WebSphere. software

IBM

WebSphere Application Server V7
Competitive Migration Guide

- Learn migration strategy and planning
- Experiment with the Migration Toolkit
- Review migration case studies

Joao Emilio S B da Silva
Kiran Mantripragada
Vasfi Gucer
Kurtcebe Eroglu
Hamdy Eed
Fabio Xavier Albertoni

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Application Server V7: Competitive Migration Guide

August 2010

SG24-7870-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xix.

First Edition (August 2010)

This edition applies to IBM WebSphere Application Server Version 7.0.

© Copyright International Business Machines Corporation 2010. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures xi

Tables xv

Examples xvii

Notices xix
Trademarks xx

Preface xxi
The team who wrote this book xxi
Now you can become a published author, too! xxiii
Comments welcome xxiii
Stay connected to IBM Redbooks xxiv

Chapter 1. Introduction 1
1.1 Objectives 2
1.2 Scope of this book 2
1.3 Applications covered in this book 3
1.3.1 Java SE and EE technologies covered 4
1.4 What is not covered in this book 4
1.5 How to use this book 5

Chapter 2. Common migration issues 7
2.1 Java EE application server compatibility 8
2.1.1 Differences in Java EE implementations 8
2.1.2 Using vendor-specific features 12
2.1.3 Deployment descriptors 12
2.2 Application portability 15
2.2.1 Application packaging 15
2.2.2 Java source code 18
2.2.3 Use of native code 18
2.2.4 Database-related issues 19
2.2.5 Java EE application clients 20
2.3 J2EE 1.4 to Java EE 5 migration considerations 21
2.3.1 Java Message Service 21
2.3.2 Java Server Pages 22
2.3.3 Servlets 22
2.3.4 Java Server Faces 22

© Copyright IBM Corp. 2010. All rights reserved.

2.3.5 Web services . . . 22
2.4 Runtime migration issues . . . 23
2.4.1 Migrating other products at the same time . . . 23
2.4.2 Resource definitions . . . 24
2.4.3 Development environment issues . . . 26

Chapter 3. WebSphere overview . . . 27
3.1 IBM WebSphere Platform . . . 28
3.1.1 Application infrastructure . . . 28
3.1.2 Application integration . . . 29
3.1.3 Business process management . . . 29
3.1.4 Business rule management systems . . . 29
3.1.5 Optimization . . . 30
3.1.6 Portals . . . 30
3.2 WebSphere Application Server V7 . . . 30
3.2.1 Overview . . . 30
3.2.2 Packaging . . . 33
3.2.3 Java Platform Enterprise Edition (Java EE) . . . 36
3.2.4 Evolving Java application development standards . . . 38
3.2.5 Supported hardware, platforms, and software . . . 39
3.2.6 New features and enhancements in WebSphere Application Server V7 . . . 43
3.2.7 Feature packs . . . 55
3.3 Development and deployment tools . . . 63
3.3.1 Rational Application Developer for Assembly and Deploy V7.5 . . . 64
3.3.2 Rational Application Developer for WebSphere Software V7.5 . . . 66
3.3.3 WebSphere Rapid Deployment . . . 67
3.3.4 Choosing which tools to use . . . 68

Chapter 4. Migration strategy and planning . . . 69
4.1 Migration process overview . . . 70
4.1.1 Types of migrations . . . 71
4.1.2 Important aspects of migrations . . . 72
4.1.3 Migration considerations . . . 73
4.2 Before beginning . . . 75
4.2.1 Migration assessment . . . 75
4.3 Migration planning . . . 77
4.3.1 Roles and responsibilities . . . 77
4.3.2 Getting help . . . 77
4.3.3 Migration project plan . . . 78
4.4 Migration implementation . . . 79
4.4.1 Implementation considerations . . . 79
4.4.2 Implementation phases . . . 80

4.4.3 Application migration 82
4.4.4 Runtime migration 83
4.5 Post deployment 83
4.5.1 Performance tuning and optimization 84
4.5.2 WebSphere Application Server V7: New opportunities 84
4.6 Summary 86

Chapter 5. Installation and configuration 87
5.1 Hardware and software 88
5.2 IBM Rational Application Developer for WebSphere Software 88
5.2.1 Installing the latest fixpack 90
5.2.2 Verifying installation 93
5.3 IBM Application Migration Tool 95
5.3.1 Installing Application Migration Tool into IBM Rational Application Developer for WebSphere Software 95
5.4 IBM DB2 Server Express Edition 97

Chapter 6. Migration from Oracle WebLogic 99
6.1 Introduction 100
6.2 Prerequisites and assumptions 101
6.3 Oracle WebLogic Server 9.2 installation 102
6.4 Application Migration Tool overview 103
6.5 Migrating Trade 3.1 for Oracle WebLogic Server 9.2 104
6.5.1 Migration approach 105
6.5.2 Configuring the initial environment 106
6.5.3 Migrating the sample application 113
6.5.4 Summary 165
6.6 xPetstore EJB migration 166
6.6.1 Migration approach 166
6.6.2 Configuring the initial environment 167
6.6.3 Migrating the sample application 175
6.6.4 Summary 197

Chapter 7. Migrating from JBoss 199
7.1 Migrating from older versions 200
7.2 Preparing the environment 200
7.2.1 Installing and configuring Apache Ant 200
7.2.2 Installing and configuring Apache Maven 201
7.2.3 Installing and configuring JBoss Application Server 202
7.2.4 JBoss Seam Framework 203
7.3 Migrating JSF-EJB3 Sample Application from JBoss 203
7.3.1 Migration approach 204
7.3.2 Verifying JSF-EJB3 sample application 204
7.3.3 Importing the EAR file and source code to RAD 205

7.3.4 Analyzing and fixing migration problems 211
7.3.5 Managing additional runtime dependencies 212
7.3.6 Checking the project using Application Migration Tool 220
7.3.7 Build and run application on integrated test environment 222
7.3.8 (Optional) Specifications level migration 222
7.3.9 Migrating to WebSphere built-in JPA provider (optional) 224
7.3.10 Summary 225
7.4 Migrating a Seam application generated with seam-gen 226
7.4.1 Migration approach 227
7.4.2 Preparing database schema 227
7.4.3 Generating Seam application using seam-gen 230
7.4.4 Verifying generated seam-gen application 234
7.4.5 Importing the EAR file and source code to RAD 235
7.4.6 Analyzing and fixing problems 240
7.4.7 Summary 253
7.5 Migrating the Online Brokerage application 254
7.5.1 Migration approach 254
7.5.2 Building the Online Brokerage application 255
7.5.3 Creating and populating the database 256
7.5.4 Importing the EAR file and source code to RAD 258
7.5.5 Analyzing and fixing the problems 261

Chapter 8. Migrating from Apache Tomcat 275
8.1 Introduction 276
8.2 Prerequisites and assumptions 276
8.3 Software installation 277
8.3.1 Apache Tomcat 278
8.4 MvnForum migration 279
8.4.1 Migration approach 279
8.4.2 Configuring the source environment 280
8.4.3 Migrating the MvnForum application 282
8.5 Easy JSP Forum migration 289
8.5.1 Migration approach 289
8.5.2 Configuring the source environment 289
8.5.3 Migrating the Easy JSP Forum application to WebSphere 291

Chapter 9. Migrating from GlassFish 301
9.1 Introduction 302
9.2 Prerequisites and assumptions 302
9.3 Java Pet Store Reference application 303
9.3.1 Installing and configuring GlassFish V2.1.1 304
9.3.2 Building and deploying Java Pet Store on GlassFish 305
9.3.3 Migration to WebSphere Application Server 306

9.4 CalculatorWS: A sample web services application 318
9.4.1 Running CalculatorWS on NetBeans 6.8 and GlassFish 3 319
9.4.2 Migration to WebSphere Application Server 320
9.5 Major issues when migrating web services applications 322
9.6 Summary 324

Appendix A. Development practices for portable applications 325
Java EE development practices 326
J2EE, standards and proprietary solutions 326
Conform to standards as much as possible 327
Abstraction is a key point 327
Develop the application architecture into layers 327
Write and maintain automated tests 328
Documentation 329
Use MVC 331
Do not reinvent the wheel 334
Develop to the specifications, not the application server 334
Build what you know 335
Always use session facades whenever you use EJB components 336
Use stateless session beans instead of stateful session beans 336
Use container-managed transactions 337
Prefer JSPs as your first choice of presentation technology 337
Enable session persistence 338
Take advantage of application server features that do not require your code to be modified 338
Play nice within existing environments 339
Embrace Java EE, do not fake it 339
Plan for version updates 340
Log your program state using a standard logging framework 340
Always clean up after yourself 340
Follow rigorous procedures for development and testing 341
Software Engineering and general development practices 341
Design patterns 341
Iterative development 344
Object Oriented (OO) approach to design and programming 347
End-to-end life cycle 350
The Rational Unified Process (RUP) 351
Agile development 353

Appendix B. Migration questionnaires 355
Business requirements 356
General information 356

Application architecture 357
Dependencies 357
Persistence 358
National language 358
Code 359
Java 359
EJB usage 359
Servlets and JSPs 360
Web services 361
Database access 361
JMS 362
JNDI naming 362
Application trace and logging 363
Struts 363
Transactions 363
Threads 364
Sockets 364
XML 364
Development migration questionnaire 365
Workstation configuration 365
Integrated Development Environment 365
Development test configuration 366
Software development skills 366
Development methodology 367
Build and packaging 367
Ant 367
Runtime migration questionnaire 368
General 368
Current hardware 368
Software 369
HTTP Server 370
Network edge 370
Availability 371
Rollout issues 371
Administration 372
Security 372
Testing migration questionnaire 373
Hardware 373
Practices and tools 373

Appendix C. Additional material 375
Locating the web material 375
Using the web material 376

System requirements for downloading the web material 376
How to use the web material 376

Related publications 377
IBM Redbooks 377
Online resources 378
How to get Redbooks 380
Help from IBM 380

Index 381

Figures

2-1 WebSphere Application Server v7 classloader in default configuration . . 10
2-2 Recommended packaging structure of Java EE application. 16
3-1 Basic architecture of an application server 31
3-2 Business application services position in SOA reference architecture . . . 32
3-3 Packaging structure of WebSphere Application Server V7.0 33
3-4 Web services run time started. 47
3-5 Multiple base profiles 48
3-6 Administrative agent 49
3-7 Administrative agent managing multiple application servers 50
3-8 Job manager management model 52
3-9 Rational Application Developer packaging 64
3-10 Rational Software Delivery Platform overview 67
4-1 Migration process overview. 71
5-1 IBM Installation Manager preferences dialog after adding repositories . . 92
5-2 RAD version information as displayed in About window. 93
5-3 Integrated testing environment server view 94
5-4 Welcome page of test environment showing current version 94
5-5 Plug-in file 96
5-6 Selecting toolkit update site 96
5-7 Verifying installation of toolkit 97
6-1 Trade3 J2EE components. 105
6-2 EAR file import 114
6-3 Trade module selections 115
6-4 Application Migration Tool configuration 117
6-5 Migration tool workspace scope settings 118
6-6 Migration Tool rule settings 119
6-7 Software Analyzer View 120
6-8 Java Code Review tab 121
6-9 WebLogic specific packages. 123
6-10 Trade web application package 124
6-11 Setting Java Build Path 125
6-12 Problems View 126
6-13 Quick Fix Wizard 127
6-14 Project Clean dialog 128
6-15 Migration tool analysis. 129
6-16 XML File Review tab 130
6-17 Quick Fix Preview 131
6-18 EJB JNDI Quick Fix. 132

© Copyright IBM Corp. 2010. All rights reserved.

6-19 WebSphere Binding file. 133
6-20 WebLogic deployment descriptor 135
6-21 Specify DB type. 135
6-22 XML File Review tab 136
6-23 WebLogic Query Language construct. 136
6-24 Convert CMP Mappings 137
6-25 CMP napping file. 138
6-26 Database Schema files 138
6-27 Show unmapped objects. 139
6-28 WebLogic Query Language error 140
6-29 WebLogic EJB deployment descriptors 141
6-30 WebLogic web deployment descriptor 142
6-31 J2EE Migration Wizard 143
6-32 J2EE version selection 144
6-33 Migration Completion dialog 145
6-34 Creating Service Integration Bus 147
6-35 Service Integration Bus. 148
6-36 WebSphere scope selection 150
6-37 MDB Activation Spec Configuration 155
6-38 Add and Remove Projects 162
6-39 Trade Application welcome page 163
6-40 Trade Application Configuration options 164
6-41 Import EAR file 176
6-42 Importing xPetstore Application 177
6-43 Database mapping generated with Top-down option 180
6-44 Migration toolkit JSP Code Review. 182
6-45 Refactoring JSP file fragments 183
6-46 XML File Review tab 183
6-47 WebLogic specific Query Language tag 184
6-48 EJB Deployment Descriptor Editor 193
6-49 Resource reference page in the EJB Deployment Descriptor editor. . . 195
7-1 Importing an EAR file using import wizard 206
7-2 Initial project structure created by the Import wizard. 207
7-3 Before and after we replace class files with respective source files in EJB module 208
7-4 Importing source files for EJB module 209
7-5 Before and after we replace class file with respective source file in web module 210
7-6 Adding a dependency to the web module. 211
7-7 Application module after importing additional dependencies 214
7-8 EJB module dependencies 215
7-9 Persistence XML Editor warning message 216
7-10 Configuring class loading for third party persistence providers. 219

7-11 Software Analyzer configuration with JBoss migration rule set 220
7-12 Migration Tool XML file review on JSFEJB3 221
7-13 Migration Tool XML file review on JSFEJB3 221
7-14 Target specification level selection 223
7-15 Specifications level migration summary 224
7-16 Before and after Specification Migration Wizard run 224
7-17 Creating test database using DB2 Command Editor 229
7-18 EAR import wizard 235
7-19 Importing myproject.ear 236
7-20 Selecting EAR Modules during import 237
7-21 EJB module structure before and after source code import 238
7-22 Importing source files 239
7-23 Before and after libraries in EAR moved to shared library 240
7-24 Shared library properties 242
7-25 EJB module class path 243
7-26 Web module dependencies 246
7-27 JDBC provider settings 248
7-28 Selecting provider for data source 249
7-29 Data Source details 250
7-30 Resource properties for data source 251
7-31 Data Sources panel of WebSphere Application Server deployment . . . 252
7-32 Creating brkrgdb database for sample application 258
7-33 brokerage_WEB module before and after source code import 259
7-34 Importing source code files to brokerage_WEB module 260
7-35 Adding dependency to HAR resources 262
7-36 JDBC provider settings 266
7-37 Data Source details 267
7-38 Data Sources panel of WebSphere Application Server Deployment . . 268
7-39 Resource references for brokerage_WEB 269
7-40 Creating a new utility module 270
7-41 Importing source files to utility project 271
7-42 Utility module contents 271
7-43 Module dependencies of web module 272
8-1 MvnForum deployed and started 288
8-2 Files selection from source_home. Do not select the forum folder. 296
8-3 Easy JSP Forum deployed and started 299
9-1 Java Pet Store migration steps overview 304
9-2 Creating “petstore” database in DB2 307
9-3 Creating tables and populating “petstore” database. 308
9-4 Select the workspace folder according to your environment 309
9-5 Resetting the RAD perspective to a Java EE Project look and feel 310
9-6 Creating a dynamic web project #2 311
9-7 Creating a dynamic web project #2 312

9-8 Package name problem when importing to RAD. 313
9-9 Create a Resource Reference for a JDNI name 315
9-10 Sample web services application development and migration steps . . 319
9-11 Select the workspace folder according to your environment. 320
9-12 Resetting the RAD perspective to a Java EE Project look and feel . . . 321
9-13 Typical infrastructure for web services applications 323
A-1 Typical Java EE MVC implementation 332
A-2 Rational Unified Process overview 352

Tables

2-1 Java and vendor deployment descriptors and configuration files 14
3-1 Supported operating systems for WebSphere Application Server V7.0 . . 40
3-2 List of databases that are supported by WebSphere App. Server V7. . . . 42
3-3 Dependency Injection 45
3-4 Web services development 45
3-5 Specification level support 65
6-1 Data source values 109
6-2 Database Connection Properties Values 109
6-3 JMS Server values 110
6-4 JMS Queue values 110
6-5 JMS Topic values 111
6-6 JMS Topic values 111
6-7 Queue connection factory values 111
6-8 Topic connection factory values 112
6-9 JMS queue connection factory configuration 151
6-10 JMS topic connection factory configuration. 151
6-11 JMS queue configuration 152
6-12 JMS queue configuration 152
6-13 JMS Topic configuration 153
6-14 JMS activation specification configuration 154
6-15 JMS activation specification configuration 154
6-16 OrderProcessor MDB's WebSphere binding configuration 155
6-17 Mailer MDB's WebSphere binding configuration. 156
6-18 JDBC provider configuration 157
6-19 JDBC data source configuration 157
6-20 J2C Authentication Alias Configuration 158
6-21 WebSphere bindings for the Account EJB 159
6-22 Trade EJB JNDI Names 159
6-23 EJB reference names 160
6-24 Default data source configuration 161
6-25 Trade web application reference JNDI names 161
6-26 JDBC Data Source configuration 169
6-27 Database Connection Properties 169
6-28 JMS Server configuration 170
6-29 Order MDB's JMS queue configuration. 170
6-30 Mail MDB's JMS queue configuration 171
6-31 JMS connection factory configuration 171
6-32 JavaMail session configuration 171

© Copyright IBM Corp. 2010. All rights reserved.

6-33 Database configuration for wizard . . . 179
6-34 JavaMail session configuration . . . 186
6-35 JDBC provider configuration . . . 187
6-36 JDBC data source configuration . . . 187
6-37 J2C Alias Configuration . . . 188
6-38 JMS queue connection factory configuration . . . 190
6-39 Mail queue configuration . . . 190
6-40 Order queue configuration . . . 191
6-41 Order MDB's JMS activation specification configuration . . . 191
6-42 Mailer MDB's JMS activation specification configuration . . . 192
6-43 OrderProcessor MDB's WebSphere binding configuration . . . 193
6-44 Mailer MDB's WebSphere binding configuration . . . 194
6-45 WebSphere bindings for the Account EJB . . . 194
6-46 Default data source configuration . . . 195
6-47 Petstore EJB's WebSphere bindings . . . 196
6-48 Order EJB's WebSphere bindings . . . 196
6-49 OrderProcessor EJB's WebSphere bindings . . . 196
6-50 OrderProcessor EJB's WebSphere bindings . . . 196
6-51 Mailer EJB's WebSphere bindings . . . 196
7-1 WebSphere Application Server V7 default binding patterns . . . 212
7-2 Additional runtime libraries . . . 213
7-3 .Shared dependencies to be copied . . . 241
7-4 Required libraries . . . 261
8-1 JDBC provider values . . . 285
8-2 Data Source values . . . 285
8-3 Database specific properties . . . 286
8-4 J2C alias configuration . . . 286
8-5 Installation values . . . 287
8-6 JDBC provider values . . . 294
8-7 Data Source values . . . 294
8-8 Database specific properties . . . 295
8-9 J2C alias configuration . . . 295
8-10 Installation values . . . 298

Examples

2-1 Web.xml excerpt 14
2-2 WebSphere Application Server V7 deployment descriptor excerpt 14
2-3 Thin client looking up an EJB deployed on a single server 20
6-1 Excerpt from the startWebLogic.cmd configuration file 108
6-2 xPetstore database configuration file 172
6-3 Database error 181
6-4 Updated INSERT statement 181
7-1 Libraries to be copied 202
7-2 Replacement grammar information 217
7-3 Code snippet 228
7-4 Seam setup command interactive output 230
7-5 Code snippet 245
7-6 256
7-7 Changes to the file 263
8-1 Database creation commands 280
8-2 Changing mvncore.xml file code snippet 281
8-3 Define data source in mvncore.xml 283
8-4 Define paths and trusted domains in mvncore.xml 284
8-5 Creating the database and the tables 292
8-6 Populate the table Members 293
8-7 Adding the required packages to access the data source 296
8-8 Changing the connection parameters 297
8-9 Remove the square brackets in the following SQL syntaxes 297
9-1 GlassFish installation commands 305
9-2 GlassFish initialization commands 305
9-3 Unpack commands 305
9-4 Folder structure for source code 314
9-5 Original index.jsp raises an “Unsupported Operation” exception due to read-only result list from JPA 316
9-6 Modified index.jsp 316
9-7 Original CatalogXmlAction.java (line 70) 317
9-8 New CatalogXmlAction.java (line 70) 317
9-9 Output messages after running the Client Consumer application for Calculator Web Services 322

© Copyright IBM Corp. 2010. All rights reserved.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2010. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX 5L™	FileNet®	Rational®
AIX®	IBM®	Redbooks®
CICS®	ILOG®	Redpaper™
ClearCase®	IMS™	Redbooks (logo) ®
CloudBurst™	Informix®	SoDA®
DataPower®	iSeries®	System i®
DB2 Connect™	Lotus®	System z®
DB2 Universal Database™	Parallel Sysplex®	Tivoli®
DB2®	Passport Advantage®	WebSphere®
developerWorks®	POWER®	z/OS®
Domino®	Rational Rose®	zSeries®

The following terms are trademarks of other companies:

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

AMD, AMD Opteron, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

FileNet, and the FileNet logo are registered trademarks of FileNet Corporation in the United States, other countries or both.

Novell, SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication helps you plan and execute the migration of J2EE applications developed for Oracle WebLogic Server, JBoss, GlassFish, and Apache Tomcat, so that they run on WebSphere® Application Server V7.

This book provides detailed information to plan migrations, suggested approaches for developing portable applications, and migration working examples for each of the platforms from which we migrated.

It is not our intention to provide a feature-by-feature comparison of these application servers versus WebSphere Application Server V7, or to argue the relative merits of the products, but to produce practical technical advice for developers who have to migrate applications from these vendors to WebSphere Application Server V7.

The book is intended as a migration guide for IT specialists who are working on migrating applications written for other application servers to WebSphere Application Server V7.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Joao Emilio Santos Bento da Silva is a Senior IT Specialist working for IBM acting as Web Middleware Security Coordinator for the IBM Global Account. He has 13 years of professional experience, eight dedicated to the Java™ platform working as developer, architect, consultant and WebSphere technical support level 3 at companies in Latin America and the United States. Emilio holds Java certifications including Sun Certified Enterprise Architect and IBM OOAD and WebSphere Application Server since version 5.

Kiran Mantripragada is a Certified IT Specialist, performing the IT Architect Job Role at the Solution Design Center of Excellence (SDCoe) in Brazil, where he is a member of the Technology Leadership Council (an IBM Academy Affiliate). He has more than 15 years of experience in IT, gaining experience in Software Development (including C++ and Java), Systems and Servers Support, Web Technologies, Application Integration and Middleware and Computer Vision research. He is Mechatronics Engineer and holds a Masters Degree in Software Engineering with research focus on Computer Vision and Pattern Recognition, from Instituto de Pesquisas Tecnológicas in Sao Paulo, Brazil.

© Copyright IBM Corp. 2010. All rights reserved.

Vasfi Gucer is a project leader at the ITSO Austin/TX. He has more than 15 years of experience in Systems Development and Enterprise Systems Management. He writes extensively and teaches IBM classes worldwide on Tivoli® software. Vasfi is also an IBM Certified Senior IT Specialist, PMP, and ITIL® Expert.

Kurtcebe Eroglu is an Advisory IT Specialist working for IBM Software Services for WebSphere (ISSW) in Turkey. He has nine years of experience in IT, focused on software quality assurance, OO design and development, JEE and EAI. Prior to IT, he designed and erected waste water treatment and industrial water demineralization plants. He has BSc degree in Environmental Engineering and MSc degrees in Engineering Management and Computer Engineering from Middle East Technical University.

Hamdy Eed is a Certified IT Specialist and a member of the Americas ITS Certification Board. He is a Senior WebSphere consultant with more than 12 years of experience consulting on WebSphere-based products, industry standards, and Application and Integration Middleware. Hamdy is also an expert in Business Process Management, WebSphere Performance tuning and WebSphere competitive J2EE migrations. He is currently a Senior Consultant for IBM Software Services for WebSphere organization. Hamdy holds a degree in Computer Science from Richards Stockton College of New Jersey.

Fabio Xavier Albertoni is a Senior IT Specialist working in Integrated Technology Delivery SSO, in Hortolandia, Brazil. He has 12 years of experience in the IT and banking industries. He has spent the last eight years developing and implementing integrated solutions using WebSphere Application Server and MQ-Series. He holds a degree in Data Processing from FATEC University of Ourinhos and a Masters degree in Computer Engineering from Instituto de Pesquisas Tecnologicas of Sao Paulo, Brazil.

Thanks to the following people for their contributions to this project:

Erica Wazewski
International Technical Support Organization, Poughkeepsie Center

Andrew Hoyt, Ed McCabe, Tom McManus, Donald Vines
IBM USA

Serdar Gere
IBM Turkey

Andrea Bauer
IBM Canada

Thanks to the authors of the previous editions of this book.

- Authors of the *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690:

  Hernan Cunico, Leonardo Fernandez, Christian Hellsten and Roman Kharkovski

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts

- Follow us on twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

This IBM Redbooks publication is the latest in a series on migrating Java Enterprise Edition (EE) applications from application servers to IBM WebSphere Application Server. The books in the series are as follows:

- *Migrating WebLogic Applications to WebSphere Advanced Edition*, SG24-5956, focuses on migration applications from WebLogic V5.1 to WebSphere 3.5.
- *Migrating WebLogic Applications to WebSphere Advanced Edition V4*, SG24-6179, discusses WebLogic V6.1 and WebSphere V4.0.
- IBM Redpaper™ *Migrating WebLogic Applications to WebSphere V5*, REDP-0448, covers migrating applications from WebLogic V7 to WebSphere V5.0.
- *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690, focuses on migrating applications from these platforms to WebSphere V6.

This book covers migration from Oracle WebLogic 9.2, JBoss 4.x and 5, Apache Tomcat 6.0.26, and GlassFish 2.1 to WebSphere Application Server V7.

This chapter summarizes the contents of this book. It also describes the process used to identify migration issues and write this document.

© Copyright IBM Corp. 2010. All rights reserved.

1.1 Objectives

Migration of Java EE applications from one application server to another is an activity that can vary from extremely simple to extremely complex. This book does not offer all possible solutions or cover every issue that can arise in a migration, but it is a good place to begin. This book provides information to build a project plan containing the most common migration-related activities and information about migrating from other application servers to WebSphere Application Server V7. The examples for WebLogic and JBoss use the Migration Toolkit for Rational® Application Developer V7 to minimize migration efforts.

This book has the goal of being a practical guide for Java EE professionals. We identify specific migration issues and make specific considerations regarding the following issues:

- Migration process
- Migration common issues
- Migration tools

1.2 Scope of this book

This book is organized around the migration of a set of Java EE sample applications. At the time of writing, the latest version of Java EE is 5. Java EE 6 is in development.

The following products are used to demonstrate migrations:

- Apache Tomcat 6.0.26
- JBoss 4.2 and JBoss 5.1
- Oracle WebLogic 9.2
- GlassFish 2.0

The main idea behind our choices is to show migration scenarios of current application servers available to IBM WebSphere Application Server V7. Each application server chosen has its own way to implement the Java EE specification, which leads us to common issues discovered and specific issues related only to a single application server.

Creation of the book is based on the experience over several years of IBM specialists involved on migrations around the world. The information presented in this IBM Redbooks is based on articles posted on Developerworks web site, meetings with senior migration specialists, IBM documentation and migration experiences of the authors of this book.

This book contains key information about WebSphere Application Server V7, information about building a project plan, common migration issues discovered, and consolidated and specific chapters demonstrating migration from each application server in scope.

Migrations have been done with various approaches. Certain migrations used Rational Application Developer V7.5. Other migrations preserved the original development and build environment of the source application. The following elements were documented in these processes:

- Step-by-step migration process
- Specific use of tools
- Migration issues we identified along the way

As a last step, migration issues we encountered were also analyzed, categorized, and expanded upon.

1.3 Applications covered in this book

In addition to authoring a few of the applications covered in this book during the project, we have selected several applications from various sources (such as sourceforge.net and IBM Performance Lab). These applications exploit several Java features (such as EJB 2.1, web services JAX-RPC, web services JAX-WS, JMS and so forth).

As many technologies, features and functionalities for each of those platforms and samples were covered as possible with the time and resources available.

Section 1.3.1, “Java SE and EE technologies covered” on page 4 discusses the Java EE technologies, features, and functionalities covered in this book. This is not a complete list of technologies available for these platforms or applications. They are used as a guide to map migration results with your own scenario.

1.3.1 Java SE and EE technologies covered

The following Java SE and EE technologies are covered in this book:

- Java Server Pages (JSP)
- Servlet
- Java Server Faces (JSF)
- Enterprise Java beans (EJB) 2.0, 2.1 and 3.0
- Java Persistence API (JPA), Open JPA
- Web services - JAX-RPC and JAX-WS
- Java Message Service (JMS)
- Java Database Connectivity (JDBC)
- Java Mail
- JSP taglibs

No problems were encountered with specific frameworks being used by the applications migrated, thus this information does not add value here. In any case, sample applications are downloadable and you can check what frameworks they are using.

All migration instructions in this book are written for Microsoft® Windows® platforms, but are not platform-specific. The same principles apply to Linux®, UNIX®, iSeries®, z/OS® and any other platform running your J2EE application.

1.4 What is not covered in this book

During the development of this book, time and resource limitations did not allow this book to cover the following areas of technology:

- Java EE Connector Architecture (JCA)
- Service Component Architecture (SCA)
- Bean-Managed Persistence (BMP) EJB
- Java Management Extensions (JMX)
- Administrative scripting
- Timer and startup services
- Clustering, server administration and configuration
- Database and operating system migration

The main focus of the book is migration and it is assumed that you are familiar with Java EE concepts, the Java language, the platforms from which you are migrating, and the development tools. Platform-specific migration chapters contain references to additional documentation as needed. A complete list of all related documentation can be found in “Related publications” on page 377.

1.5 How to use this book

It is advised to read the entire book, but depending on time and interest, each chapter can be read separately. The book contains the following chapters.

- Chapter 2, “Common migration issues” on page 7 discusses the migration issues discovered in the writing of this book. It also discusses common pitfalls found in typical applications and suggests ways to solve them.
- Chapter 3, “WebSphere overview” on page 27 discusses WebSphere Application Server editions, operating systems, and supported databases. It includes an overview of the Rational Application Developer V7.5 development tool. It also contains a section discussing new features for both products.
- Chapter 4, “Migration strategy and planning” on page 69 provides a concise migration process overview, related main activities, and considerations about how to go through migration of your environments.
- Chapter 5, “Installation and configuration” on page 87 is a guide to install and configure IBM products used in this book (such as DB2® Universal Database™ 9.2, Rational Application Developer V7.5, and the Migration Toolkit).
- Chapter 6, “Migration from Oracle WebLogic” on page 99 presents migration examples from Oracle WebLogic 9.2 to WebSphere Application Server V7. We have migrated two applications (Trade and xPetstore EJB) using Rational Application Developer V7.5 along with the new Migration Toolkit.
- Chapter 7, “Migrating from JBoss” on page 199 presents step-by-step migration examples of three applications from JBoss to WebSphere Application Server with two alternatives:
  - Maintaining the development environment
  - Migrating to Rational Application Developer V7.5 with the migration toolkit.
- Chapter 8, “Migrating from Apache Tomcat” on page 275 presents step-by-step migration examples of two applications from Apache Tomcat 6.0.26 to WebSphere Application Server V7 without using the IDE.
- Chapter 9, “Migrating from GlassFish” on page 301 presents step-by-step migration examples of two applications, including a web services application built on from GlassFish 2.1 to WebSphere Application Server V7 using Rational Application Developer V7.5.
- Appendix A, “Development practices for portable applications” on page 325 contains guidance on portability and development, covers popular design patterns and gives a brief overview of the Rational Unified Process (RUP).
- Appendix B, “Migration questionnaires” on page 355, provides a set of migration questionnaires with sample questions that helps you build your own migration questionnaire during the migration assessment.

2

Common migration issues

This chapter discusses the most common migration issues the reader might come across when migrating from other Java Enterprise Edition (EE) platforms to WebSphere Application Server V7.

Although the idea behind Java and Java EE is portability and "Write Once, Run Anywhere (WORA) functionality, this is not always the case, because the unique way in which each vendor implements the Java EE specification often leads to problems when migrating a Java EE application. Vendors also implement features that are not included in the Java EE specification. This is why Sun provides a list of vendors with Java EE-compatible products. Testing is used by Sun to verify that a Java EE application server complies with the specification. This ensures that applications deployed on one Java EE application server also run on another vendor's application server.

Java EE applications can also conflict with the Java EE specification. The Java EE specification is so complex that it is difficult for developers to know all the details. Tools such as Rational Application Server V7.5 can be used to verify that an application is written according to the specifications.

This chapter provides a list of migration issues caused by the reasons mentioned. This chapter has the following sections:

- "Java EE application server compatibility" on page 8
- "Application portability" on page 15
- "J2EE 1.4 to Java EE 5 migration considerations" on page 21
- "Runtime migration issues" on page 23

© Copyright IBM Corp. 2010. All rights reserved.

2.1 Java EE application server compatibility

Although the Java EE specification covers an increasingly wider spectrum of enterprise applications development, there is still the need for vendor-specific features.

Vendor-specific features used without proper planning cause migration problems. Small differences or bugs in the vendor's implementation of the Java EE specification can also cause compatibility problems. These issues are summarized in the following sections:

- Differences in Java EE implementations
- Using vendor-specific features
- Deployment descriptors

2.1.1 Differences in Java EE implementations

This section describes the differences the authors encountered during the migration process and which are caused by differences in how the Java EE specification is implemented by vendors. The Java EE specification leaves room for interpretation, which leads to issues when migrating the application to a new application server. These interpretations are also subject to discussion and updates to the specification.

A few of the problems faced during migration were not really predictable by the developers. Others are more likely a bug or resiliency of the application server or web container implementation. To give you an example, a regular request for a JSP or Servlet using a duplicate slash "/" as in the URL `http://localhost:8080//testapp/index.jsp` works in Apache Tomcat 6 and Glassfish 2 but does not work in WebSphere Application Server V7.

In another case, the authors found an incompatibility of JPA implementations between Glassfish and WebSphere Application Server implementations. Retrieving a list using EntityManager and then sorting that list using a Comparator returned a message saying `Result lists are read-only`.

These and other examples require lots of testing to catch, but after you find the first, the search for other similar codes is going to be easier. Remember to document your findings. Use a common repository, so they can be brought to light and shared across the entire team. Most likely these implementation issues are discussed in a regular meeting (see Chapter 3, "WebSphere overview" on page 27).

Classloaders

Another area of interest is classloader implementations. Each vendor has its own way to implement the classloader. WebSphere Application Server V7 is 100% compliant with Java EE specifications and has additional features that allow for the coexistence of applications using various versions of the same library, for instance.

Another known problem is that vendors, as we all do, make usage of open source libraries (such as apache commons, log4j, axis and so on). JBoss, for example, has an implementation of classloader that does not differentiate classes from applications and the server itself. So, for instance, an application that uses log4j does not need to deploy log4j library. The best solution in that case is to use the exact same version of log4j that it is shipped with JBoss to avoid problems. You might face problems, such as ClassNotFound, when deploying JBoss-migrated applications. In these cases, you have couple of options. In Chapter 7, “Migrating from JBoss” on page 199 we provide solutions to overcome this problem.

For more about WebSphere Application Server V7 classloaders, see IBM Redbooks publication *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690.

WebSphere Application Server V7 classloader

The WebSphere Application Server V7 classloader consists of four components:

- System classloader

  This component is provided by the Java virtual machine. It is responsible for loading classes from the CLASSPATH, bootstrap and extensions class paths.

- WebSphere runtime classloaders

  This component loads the WebSphere Application Server V7 runtime classes. All classes in the ws.ext.dirs class path are added to this classloader.

- Application classloaders

  This component is responsible for loading resources that are part of an EAR module, for example JAR, EJB and WAR files.

- Web module classloaders

  This component loads the classes in WEB-INF/classes. It is possible to override this behavior and have the application classloader load the classes.

Each classloader is a child of the parent classloader, as illustrated in Figure 2-1. By default, when a resource needs to be loaded, all classloaders ask the parent classloader to locate it. If it cannot be found, the resource is loaded by the child classloader. If none of the classloaders can find the resource, you receive a ClassNotFoundException.

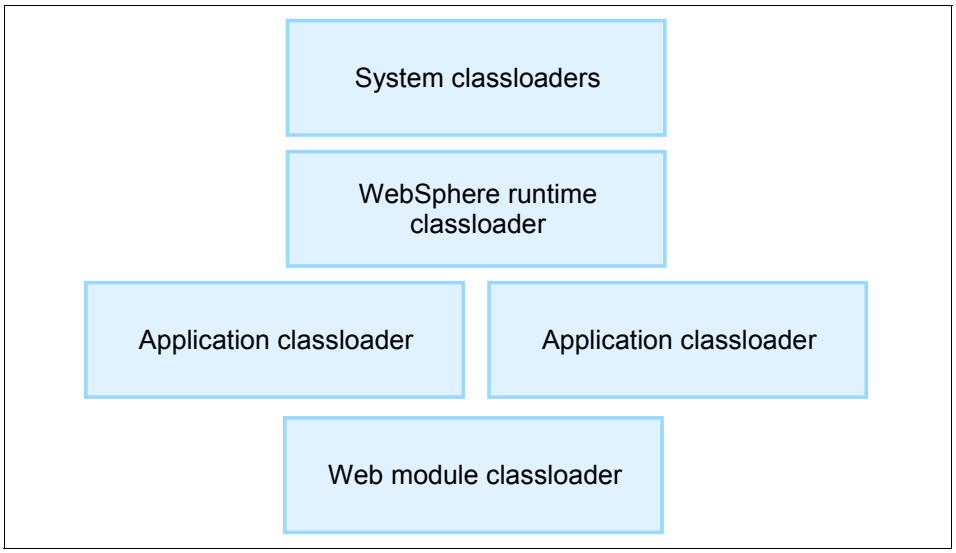


Figure 2-1 WebSphere Application Server v7 classloader in default configuration

Classloader isolation policies

The WebSphere Application Server V7 application and web module classloaders can be configured to enable various packaging schemes. The default is the multiple mode.

Application classloader

The application classloader can be configured to use the following modes.

- Single

  When in single mode, the application classloader is shared by all modules in the EAR application. For example, all JAR, WAR, and EJB modules have visibility to all classes located in all modules.

  There are limitations when using this mode, such as for JSF applications. See the following web page for more information:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cweb_javaserver_faces.html`

- Multiple

  When in multiple mode, an application classloader is created for each module in the EAR application. This means that the various modules are isolated from each other.

Tip: During the authors' migration experiences, they encountered clients who require enterprise applications to communicate across various WebSphere cells. In this case a CORBA style and an explicit JNDI name are specified. The following article presents more information about the topic:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/cejb_bindingsejbfp.html

Web module classloader

The web module loads resources from the WEB-INF/lib and WEB-INF/classes directories in the WAR file. This is the default behavior and can be overridden by setting the policy to `Application`. This makes the application classloader load the resources from WEB-INF/lib and WEB-INF/classes, instead of the web module classloader.

Classloader modes

WebSphere Application Server V7 supports two classloader modes.

- Parent first

  This is the default mode for all classloaders. It is the Java EE and Java standard. The loading of the resources is first delegated to the parent classloader. If the parent classloader is unable to find the resource, the child classloader tries to handle it.

- Parent last

  Using the parent last class loading policy makes the classloader work opposite of the parent first mode. The resource is loaded from the application or web module classloader first. If the resource is not found, the loading is delegated to the parent classloader.

Shared libraries

WebSphere Application Server V7 allows you to configure dependencies that are shared by either all applications in the server or just one specific application classloader. Shared libraries are configured using the administration console and can be specified at the application or server level.

Bundled libraries

WebSphere Application Server V7 contains open source libraries that are located in the server class path. If you are using the same libraries and there is a version mismatch, you might receive one of the following errors:

- java.lang.NoSuchMethodError
- java.lang.ClassCastException
- java.lang.NoClassDefFoundError

To fix these errors, you can try to experiment with various class loading policies, such as parent last instead of parent first.

The following list details the common libraries that might create problems:

- WebSphere Application Server V7 includes the Jakarta Commons Logging (JCL), which is known to cause problems when migrating.

  The following article describes how to solve JCL problems on WebSphere:

  http://www.ibm.com/support/docview.wss?uid=swg27004610

- JDom is also bundled with WebSphere Application Server V7.

2.1.2 Using vendor-specific features

When faced with the choice of using a vendor-specific feature, you have the option of not using it and accepting the limitations of the Java EE specification. This is good from a portability and migration point of view.

You can also use the vendor-specific feature and accept the risk and potential future migration costs. If you decide to proceed with this option, ensure that you abstract away the implementation details, for example by designing interfaces, not concrete classes.

Vendors have various additional features:

- Startup and shutdown tools
- Timer services
- Automatic primary key generation
- JDBC connection pooling

2.1.3 Deployment descriptors

The Java EE specification does not cover all the aspects of application deployment and configuration. This is why there are vendor-specific deployment descriptors. Table 2-1 on page 14 lists both the Java EE and vendor-specific deployment descriptors. Each application server uses a separate set of files. This is because they are implemented differently and have separate feature sets.

Migration of Java EE application using previous specifications such as J2EE 1.4 come with one of the biggest problems associated to deployment descriptors when migrating to WebSphere Application Server V7. The problem and solutions to the problem are summarized as follows:

- Container-Managed Persistence (CMP) to database schema mapping

  WebSphere Application Server V7 uses another way of mapping CMPs to the database schema, compared to the other application servers covered in this IBM Redbooks publication. The better way of creating these mappings is by using Rational Application Developer V7. You can use the EJB to Relational Database Synchronization (RDB) mapping in Rational Application Developer V7 to create your back-end and map your entity beans to the corresponding database table. This wizard also creates the necessary JDBC code used for accessing the database at run time.

- The Java EE standard deployment descriptors are mapped to the WebSphere Application Server V7-specific deployment descriptors with IDs, as illustrated in Example 2-1 on page 14 and Example 2-2 on page 14. This means that the ID in the WebSphere Application Server V7-specific deployment descriptor must match the one in the Java EE standard deployment descriptor.

- WebSphere Application Server V7 validates the deployment descriptors more strictly than, for example, JBoss and Oracle Weblogic. This leads to the following problems when migrating:

  - Deployment descriptors fail validation because the elements are not in the proper order or the DTD is incorrect.
  - JSPs do not compile.

- Use web.xml to map JNDI references that were placed on vendor-specific deployment descriptors. If you are migrating to Java EE 5 you are likely making use of annotations that simplify references using embedded dependency injection implemented by WebSphere Application Server V7 such as @EJB.

- Switch JNDI reference mapping since WebSphere Application Server V7 JNDI standards is different than in other vendors. It is usually different in all other vendors, so it is actually a migration issue for almost every application server to another.

Next, we show an example of how the standard J2EE web.xml deployment descriptor and WebSphere Application Server V7 are mapped with IDs. In the example, a resource reference, having the ID `ResourceRef_2`, is mapped to a JNDI name pointing to a resource that is managed by WebSphere Application Server V7. See Example 2-1 on page 14 and Example 2-2 on page 14.

Example 2-1 Web.xml excerpt

```
...
      <resource-ref id="ResourceRef_2">
         <res-ref-name>jms/QueueConnectionFactory</res-ref-name>
         <res-type>javax.jms.QueueConnectionFactory</res-type>
         <res-auth>Application</res-auth>
         <res-sharing-scope>Shareable</res-sharing-scope>
      </resource-ref>
...
```

Example 2-2 WebSphere Application Server V7 deployment descriptor excerpt

```
...
    <resRefBindings xmi:id="ResourceRefBinding_1117409057536"
jndiName="jms/QueueConnectionFactory">
        <bindingResourceRef href="WEB-INF/web.xml#ResourceRef_2"/>
    </resRefBindings>
...
```

Table 2-1 lists the Java EE standard and vendor-specific deployment descriptors.

Table 2-1 Java and vendor deployment descriptors and configuration files

Description	Java EE specification	WebSphere Application Server V7	Oracle WebLogic	JBoss	Apache Tomcat
EAR application	application.xml	ibm-application-bnd.xmi ibm-application-ext.xmi	weblogic-application.xml	jboss-app.xml	N/A
WAR application	web.xml	ibm-web-bnd.xmi ibm-web-ext.xmi	weblogic.xml	jboss-web.xml	N/A
EJB application	ejb-jar.xml	ibm-ejb-jar-bnd.xmi ibm-ejb-jar-ext.xmi ibm-ejb-access-bean.xml	weblogic-ejb-jar.xml weblogic-cmp-rdbms-jar.xml	jboss.xml jaws.xml jbosscmp-jdbc.xml	N/A
J2EE client application	application-client.xml	ibm-application-client-bnd.xmi ibm-application-client-ext.xmi	client-application.runtime.xml	jboss-client.xml	N/A

Description	Java EE specification	WebSphere Application Server V7	Oracle WebLogic	JBoss	Apache Tomcat
Server configuration	Not covered by the Java EE specification	Mainly located in <was_home>\profiles\<profile_name>\config\cells	Mainly located in <domain_home>\config.xml	Mainly located in the <jboss_home>\server directory	Mainly located in the <tomcat_home>\conf directory and <web_app>\META-INF/context.xml

2.2 Application portability

This section describes the migration issues that affect application portability. Application portability, or the lack thereof, is perhaps the most important issue from a migration point of view.

2.2.1 Application packaging

Packaging is important from a portability and migration point of view. Each application is packaged differently. This, combined with the classloader implementations in each application server, can potentially create problems when migrating. The application might run in the source environment but not in WebSphere Application Server V7. To avoid problems, always follow the Java EE specification, which contains specifications for application packaging.

An application is usually packaged into an EAR file, which can consist of the following modules:

- EJB

  The module contains the EJB class files and deployment descriptors.

- WAR

  Web application module containing deployment descriptors.

- JAR

  A JAR (Java ARchive) file can contain common classes used by other modules and deployment descriptors for a Java EE application client.

- RAR

  Resource Adapter archives is a JAR file containing resource adapters for the J2C architecture.

Figure 2-2 illustrates the recommended way of packaging an application.

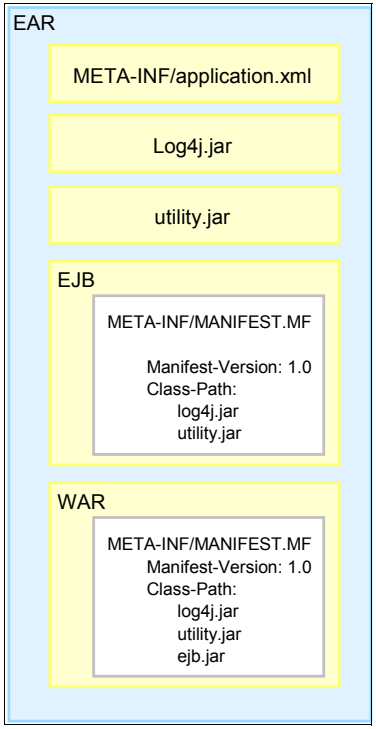


Figure 2-2 Recommended packaging structure of Java EE application

The EAR package consists of the following modules:

- EJB module

  The EJB module depends on Log4j, which is used for logging. It also depends on utility classes. Each dependency is declared in the `META-INF/MANIFEST.MF` file.

- Web application module

  The web application module depends on Log4j, the utility library and the EJB module. Each dependency is declared in the `META-INF/MANIFEST.MF` file.

> **Note:** The way you package the application might be different, depending on whether you are using Local or Remote EJB interfaces. If using Local EJB interfaces, the client must have access to all EJB classes. If using Remote interfaces, you can optionally create a JAR containing only the client view of the EJBs. Rational Application Developer V7.5 usually creates a separate project with EJB interfaces.

- Log4j.jar

  Apache Log4j is a logging implementation used throughout the application. Third-party libraries that are used by the EJB module or more than one module are placed in the application root.

- utility.jar

  This is a utility library that contains common application code. Code that is used by more than one module is located in this file.

Migration problems

Tip: Always include the version number of libraries, frameworks, and applications in either the MANIFEST.MF or the name of the file. During our migration process, we were unable to identify the version of many frameworks. Knowing the version number is critical when getting support.

During the migration of the sample applications, we found the following packaging-related problems.

- EJB module references classes in the WAR module

  The application worked on JBoss because of the JBoss Unified Classloader flat hierarchy. It stops working when deployed to WebSphere Application Server V7, because the EJB cannot view what the WAR module contains.

  Tip: The EJB module should not depend on classes in the web module. This creates a dependency that does not work on platforms other than JBoss. Extract these classes from the web application module and put them in a utility JAR. Use the `META-INF/MANIFEST.MF` file to declare the dependency.

- EJB module references libraries located in the WAR module

  In one application, EJBs imported classes from both Apache Axis and Apache Struts. The required JAR files were located in the web modules `WEB-INF/lib` folder and only visible to that module.

  Tip: Dependencies on libraries that are shared between Java EE modules in an application are in the root of the EAR file. The modules that have the dependencies should declare the dependency by using the `META-INF/MANIFEST.MF` file.

  You can place common JARs in a shared library in WebSphere and set the scope to **Application**. All modules in the EAR file are able to use such libraries. Shared libraries can also be set at the server scope.

2.2.2 Java source code

Many of the issues related to the Java source code are related to specify vendor's implementations of specifications. Here is a list of changes that you might have to perform:

- JNDI-related:
  - Context Factory provided by WebSphere: `com.ibm.websphere.naming.WsnInitialContextFactory`
  - Provider URL, for instance: `corbaloc:iiop:<hostname>:2809` (or other if changed)
  - Discard other JNDI properties specific to other application servers such as in JBoss: `java.naming.factory.url.pkgs`
- Avoid using a .jsp extension for JSP fragments, use .jspf instead. Every .jsp must compile
- Use correct case for tag/attribute names. Tag/attribute names are case sensitive

2.2.3 Use of native code

Java allows the use of native code (written in C or C++ for example) with the use of Java Native Interface (JNI). Using native code can potentially cause problems when migrating to a new operating system.

During migration assessment, discover whether or not an application is using native code. For instance:

- Libraries in windows to manage services through WMI
- Use of shell scripts capabilities in an UNIX environment
- Integration with a device
- Any other library that makes usage of native code

Sometimes migration happens in many areas including operating systems, so make the code portable after you move to the new server. Migrating applications that make usage of native code might also require a the native code to be re-compiled, for instance in a migration from a 32-bit environment to a 64-bit environment. In addition, changes in the Java code calling the JNI API might be needed as the JNI specifications can change from version to version.

2.2.4 Database-related issues

In our migration examples, we used applications that were developed for and running on various databases, including HSQL and MySQL. We migrated these to use DB2 Universal Database V9.5. This section describes the issues that we encountered and the solutions to them:

- Migrating database schemas

  Most issues with migrating database schemas were easy to solve either by manually editing the DDL or SQL scripts or running them through a tool.

  For more about MySQL migration to DB2, see the IBM Redbooks publication *MySQL to DB2 Conversion Guide*, SG24-7093.

  The following problems were the most difficult:

  - Data type overflow

    When moving one database to DB2 from MySQL, we had to change the data type of one column from DATE to TIMESTAMP, because the application was inserting TIMESTAMP.

  - Automatic primary key generation

    MySQL supports AUTO_INCREMENT columns. The DB2 equivalent for this is the *identity* column. We changed the schema to use identity columns.

- Migrating J2EE 1.4 CMPs

  In our migration examples, we had to map CMPs to the existing database schemas. We used Rational Application Developer V7.5, EJB to RDB mapping tool, which supports the following options:

  - Top down

    This creates a database schema from existing CMPs.

  - Bottom up

    By using the bottom up approach, you can generate CMPs from an existing database schema.

  - Meet-in-the-middle

    This option allows you to map the CMPs to database tables and columns automatically. The tool tries to match column and table names to the CMP. If the column or table names do not match, you have the option of creating the mapping manually with the Mapping Editor.

- Primary key generation

  Both JBoss and Oracle WebLogic contain vendor-specific EJB extensions that provide support for automatic primary key generation, as described in 2.1.2, “Using vendor-specific features” on page 12.

2.2.5 Java EE application clients

A Java EE application client is a stand-alone client running in a separate process from the application server. The client is given access to resources in the WebSphere Application Server V7 Java EE Client Container. The Java EE application client can be used to communicate with EJBs or other resources deployed on the application server. There are several use cases where Java EE application clients are appropriate:

- Alternative user interface for the application
- Running unit tests
- Administrative tasks

Migrating Java EE application clients is not covered in practice by this book. This book only provides an overview of the options provided by WebSphere Application Server V7.

WebSphere Application Server V7 Java EE application clients are executed with the following command:

```
<was_home>/bin/launchClient.bat client.ear
```

The script sets up all necessary class path and configuration variables and runs the Java EE application client, which must be packaged as an EAR file.

Thin clients

Thin application clients provide a more lightweight alternative to Java EE application clients. The thin clients access WebSphere through RMI-IIOP and do not have access to all the services provided by WebSphere.

For example, JNDI lookups do not have access to java:comp/env. Instead, the thin client must provide the fully qualified path, including the physical location, as illustrated in Example 2-3.

Example 2-3 Thin client looking up an EJB deployed on a single server

```
...
context = new InitialContext(env);
remote =
(CustomerServiceRemote)context.lookup("com.ibm.itso.CustomerServiceRemo
te");
...
```

To run thin clients, you need to install the Java EE and Java thin application client, which is part of the WebSphere Application Server V7 installation package and can be installed separately.

Refer to the WebSphere Application Server V7 Information Center for more information about how to run and develop thin clients:

http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/index.jsp?topic=/com.ibm.ws.ast.st.common.ui.doc/topics/tjavathin.html

2.3 J2EE 1.4 to Java EE 5 migration considerations

Java EE is evolving all the time, and new specifications and requirements are defined periodically with a new Java EE specification. WebSphere Application Server V7 supports all of the Java EE 5-related technologies, but you can still deploy J2EE 1.4 applications as well.

If you plan to maintain your applications, consider migrating them to Java EE 5. This way, you are up to date, and the changes in the next Java EE specifications are easier to carry out.

For more about Java EE 5 new opportunities, see IBM Redbooks publication *Experience JEE! Using Rational Application Developer V7.5*, SG24-7827.

Java EE 5 has significant changes. Developers were have improved deployment descriptors and XDoclet tags. Changes in the specification are also significant in terms of portability, which helps future migrations (unless Java EE 6 breaks paradigms). These and other changes are the whys to make Java EE specification migration and application server migration.

2.3.1 Java Message Service

Java Message Service messaging is a common issue when migrating from other vendors to WebSphere Application Server V7. Each vendor has its own way to implement specifications and there are many approaches on how to use messaging in applications.

Changes in the JMS 1.1 should preserve backward compatibility with JMS 1.0.2b and simplify and unify the programming API. Check your old application to see if you can use these new JMS 1.1 functions.

Most of the issues related to JMS are to make sure that all resource configurations available in the old environment are set up in WebSphere Application Server V7. That is an assignment for the systems administrator, who is likely receiving an education on maintaining and supporting this new environment. In 2.4, “Runtime migration issues” on page 23, we talk more about runtime migration and resource configurations.

2.3.2 Java Server Pages

JSPs might vary from vendor to vendor. WebSphere Application Server V7 is more restricted about validating DTDs and schemas and does not allow deviations (such as case sensitive names).

Another common issue with migrating JSP pages is that a few containers have implicit importation of classes in packages such as java.util and requires updating JSPs with import statements.

Other than that, migration is straightforward.

2.3.3 Servlets

Servlets are as old as Java EE implementations, which is why no issues were raised during our migration. We are going to a new Java EE implementation with such great improvements in code maintainability, that developers might not want to lose the change of use annotations that simplify their lives.

Use of new annotations (such as @servlet) helps avoid problems for teams that have been facing issues in updating deployment descriptors For those who do not like XDoclet overhead, this is a good opportunity to get rid of it.

2.3.4 Java Server Faces

Java Server Faces application migrations done during our labs did not present any issues other than the ones related to JSP pages for Java EE 5 applications. It does not means there are no issues migrating JSF applications. Each vendor has its own implementation of JSF for Java EE 5.

In previous versions of WebSphere Application Server and with other vendors, JSF applications had to import JARs into web modules or place them in the class path, such as in the JBoss lib directory. See the appropriate vendor-specific chapter for more details on how to handle those issues.

2.3.5 Web services

Migrating web services involve several areas of expertise and resource skills. During our migrations we found that there are multiple approaches for web services migrations. There are three main areas of changes to be performed:

- Development environment, migration of IDEs
- Web services clients
- Web services deployment

2.4 Runtime migration issues

Runtime migration is a full time job for migration specialists. It involves several areas, roles, and responsibilities. The following list details groups that might be interested in runtime migration:

- Support teams:
  - Operating System
  - Web Middleware
  - Database
- Test teams
- Deployment specialists
- Infrastructure architects
- DNS and Network teams
- Security groups
- Project managers
- Legacy systems
- Application focal points
- Others

There is a whole set of activities that must run in parallel with application migration:

- Installation
- Configuration of all hardware and software, operating systems, database products and so forth

Infrastructure architects must design solutions for the new environment. They need to decide which topology to implement, interfaces with other systems, compatibility checks such as operating system level versus WebSphere Application Server V7 or JDBC drivers for databases, firewall rules, SOA infrastructure and other infrastructure components related.

2.4.1 Migrating other products at the same time

Building a new environment is apparently a simple task and sometimes it is. It is not in the scope of this book to get into details of this aspect but we do present a few considerations in this matter.

Usually migrations of operating system levels are straightforward, such as migrating from AIX® 5.3 to AIX 6.1 to follow pre-requirements of WebSphere Application Server V7. In other cases (such as migrating to UNIX implementations) are more challenging because of the features that might have been used by the previous environment, a few commands and parameters might vary. Of course the more problematic migration is from Windows to UNIX and

vice-versa, which is a totally separate environment and requires changes in the application, resolution of education issues, and changes in scripts used to automate support activities.

It is the same situation when migrating other components (such as databases or messaging products). New drivers to be used, new features to discover, deprecated functions to change in both application code and resource configurations, and other features might fill gaps not covered on the previous environment.

You need to take into account all these factors when migrating from one application server to another.

2.4.2 Resource definitions

Java EE containers provide implementation of specifications that allows developers to interact with databases, systems integration bus and web services. They also provide additional configurations to set up security for application users and groups, authentication mechanisms and providers, SSL certificates, and administrative roles and functions. In the following sections we present best practices for resource definitions during an application server migration project.

JMS resources

A common issue in migrating JMS resources is that the way each application server configure JMS resources vary from vendor to vendor.

In WebSphere Application Server V7, before you can define or create JMS resources (such as queues, topics and connection factories) you need first to define a service integration bus. A service integration bus in WebSphere Application Server V7 is the back end that is used by the default messaging provider.

More info about messaging, including tutorials and samples can be found at the Information Center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/welc6tech_msg.html

Also, see *WebSphere Application Server V7 Messaging Administration Guide*, SG24-7770.

JDBC resources

Reconfiguring the JDBC resources is probably the most common migration step for any migration. Generally, migrating from application servers might also come with migration of database products or versions, requiring JDBC drivers updates, so be sure to use the appropriate JDBC driver to avoid problems with new implementations for new databases. When keeping the database make sure you are using the correct driver as well.

In WebSphere Application Server V7 there are three configurations to be performed to set up a data source:

- JDBC driver, usually ready to go for most of the most used databases
- J2C Authentication data, where it is configured the user and password to be used to connect in the database
- data source itself

For more about configuring JDBC resources, see the following web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/welc6tech_dat.html

Security resources

There are several areas to configure security resources for your migrated applications and the administrative functions and roles, such as authentication mechanisms, authorization mechanisms, security for web services, and so on.

Because this is a sensitive area of migration that probably involves security standards to be applied (such as ITCS 104 (Information Technology Security Standards, internal IBM security standards), we prefer to send the reader to the appropriate reference materials available. A few basic configurations are shown in the product-specific migration chapters of this book.

To learn more about security configurations and resources on WebSphere Application Server V7, see the following web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/welc6topsecuring.html

Also refer to *WebSphere Application Server V7.0 Security Guide*, SG24-7660.

JVM arguments, class path, and system properties

The runtime migration varies case-by-case because system administrators have their own way of managing configurations. A few use start script commands, others set up environmental variables in the process user ID profile and other specific configurations or behavior of application servers.

Here are a few tips for you to try to avoid migration issues:

- Verify your startup scripts and check for JVM arguments
- On UNIX machines, check the profile of the process ID for any related environmental configuration
- Open vendor specific configuration files (such as server.xml in JBoss and config.xml in Weblogic) to check for additional configurations
- Understand how the old application server implements class path
- Ensure that you know what JARs are extending your old application server libraries

You can find more about configuring process definitions in *WebSphere Application Server V7 Administration and Configuration Guide*, SG24-7615.

2.4.3 Development environment issues

Development environments and techniques used to compile, assemble, and deploy applications have several combinations that vary case-by-case. We have seen a few patterns of these techniques such as using vendor-specific IDEs (for example Rational Application Developer V7.5), using Eclipse with a different sort of plugins and combination of Maven, Ant, XDoclet and other IDEs.

WebSphere Application Server V7 is not dependant of any IDE as it is a 100% compliant application server according to Java EE specifications. During our migrations we used two approaches:

- Keep the existent development environment and manually make the adjustments in the configuration files, ant build files, and replacement of XDoclet tags, where applicable. Depending on which Java EE components, extensions based on previous application server and specific configurations being used by the application code and deployment descriptors, complexity of these changes might vary from a simple to a time consuming exercise.
- Import applications into Rational Application Developer V7.5 and make use of the Migration Toolkit for JBoss and Weblogic. Rational Application Developer V7.5 facilitates the migration of application servers, Java EE specifications (for example, J2EE 1.4 to Java EE 5) and known migration issues from WebLogic and JBoss to WebSphere Application Server V7. More information about how to accomplish this task with Rational Application Developer V7.5 and Migration Toolkit can be found in Chapter 6, “Migration from Oracle WebLogic” on page 99 and Chapter 7, “Migrating from JBoss” on page 199.

3

WebSphere overview

It is common to refer IBM WebSphere as WebSphere Application Server, but in fact, IBM WebSphere is a bundle of software for on demand business and service-oriented architecture (SOA) for enterprises. Providing comprehensive leadership, the IBM WebSphere platform is evolving to meet the demands of companies faced with challenging business requirements, including the following tasks:

- Increasing operational efficiencies
- Strengthening client loyalty
- Integrating disparate platforms and systems
- Responding with speed to any market opportunity or external threat
- Adapting existing business to change quickly

The IBM WebSphere platform is designed to enable you to build business-critical enterprise applications and solutions and combining them with innovative new functions. WebSphere includes and supports a wide range of products that help you develop and serve your business applications. They are designed to make it easier for clients to build, deploy, and manage enterprise applications, dynamic web sites, and other more complex solutions productively and effectively.

In this chapter, you get an overview of the WebSphere Application Server V7 and learn the packaging distribution options. We take a quick look at the new features and the main differences from earlier versions. Finally, you find information about others WebSphere products used in this IBM Redbooks publication, to perform the migrations.

© Copyright IBM Corp. 2010. All rights reserved.

This chapter has the following sections:

- "IBM WebSphere Platform" on page 28
- "WebSphere Application Server V7" on page 30
- "Development and deployment tools" on page 63

3.1 IBM WebSphere Platform

WebSphere is the IBM brand of software products designed to work together to help deliver dynamic e-business quickly. WebSphere provides solutions for a positive effect on a client's business. It also provides solutions for connecting people, systems, and applications with internal and external resources. WebSphere is based on infrastructure software (middleware) designed for dynamic e-business. It delivers a proven, secure, and reliable software portfolio that can provide an excellent return on investment.

3.1.1 Application infrastructure

With WebSphere, you can build, deploy, and run applications in a proven, secure and flexible environment.

- WebSphere Application Server Family
- WebSphere Application Server Community Edition
- WebSphere sMash
- Customer Information Control System (CICS®) Transaction Server
- CICS Tools
- WebSphere Application Server Network Deployment
- WebSphere Virtual Enterprise
- WebSphere CloudBurst™ Appliance
- WebSphere Application Server Hypervisor Edition
- WebSphere eXtreme Scale
- WebSphere Real Time
- WebSphere Extended Deployment Compute Grid

3.1.2 Application integration

Use WebSphere to link applications and services for operational efficiency and cost savings.

- WebSphere MQ
- WebSphere Message Broker
- WebSphere DataPower® Integration Appliance XI50
- WebSphere Enterprise Service Bus
- WebSphere DataPower Low Latency Appliance XM70
- WebSphere Service Registry and Repository
- WebSphere MQ File Transfer Edition
- WebSphere DataPower XML Security Gateway XS40
- WebSphere Transformation Extender Industry Packs
- WebSphere Partner Gateway
- WebSphere Adapters
- WebSphere DataPower B2B Appliance XB60

3.1.3 Business process management

With WebSphere, you can optimize business performance by documenting, deploying and continuously improving end-to-end business processes.

- FileNet® Business Process Manager
- WebSphere Business Compass
- WebSphere Business Events
- WebSphere Business Modeler
- WebSphere Business Monitor
- WebSphere Business Services Fabric
- WebSphere Dynamic Process Edition
- WebSphere ILOG® Business Rule Management Systems
- WebSphere Industry Content Packs
- WebSphere Integration Developer
- WebSphere Process Server
- WebSphere Sensor Events
- WebSphere Service Registry and Repository

3.1.4 Business rule management systems

WebSphere provides easy, safe, reliable control over automated decisions used by business systems.

- WebSphere ILOG JRules BRMS
- WebSphere ILOG Rules for .NET BRMS

3.1.5 Optimization

The optimization technologies in WebSphere combine applied mathematics and computer science to help businesses make smarter decisions.

- IBM ILOG ODM Enterprise
- IBM ILOG OPL-CPLEX Development Bundles
- IBM ILOG Core Products & Technologies

3.1.6 Portals

Deliver a point of personalized interaction with applications, content, processes and people.

- WebSphere Portal Server

Where to look for more information: To learn more about WebSphere products, see the following web page:

`http://www-01.ibm.com/software/websphere/`

3.2 WebSphere Application Server V7

WebSphere Application Server is the implementation by IBM of the Java Platform Enterprise Edition (Java EE). It conforms to the Java EE 5 specification. WebSphere Application Server is available in unique packages that are designed to meet a wide range of customer requirements. At the heart of each package is a WebSphere Application Server that provides the runtime environment for enterprise applications.

For more information about IBM WebSphere Application Server V7, see the IBM Redbooks publication *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708.

3.2.1 Overview

WebSphere Application Server is the IBM runtime environment for Java-based applications. This section gives an overview of the options and functionalities that WebSphere Application Server V7.0 offers:

- Application server purpose
- Evolving Java application development standards
- Enhanced management
- Broader integration
- Advanced tooling and extensions

Application server purpose

An application server provides the infrastructure for executing applications that run your business. It insulates the infrastructure from hardware, operating system, and the network. See Figure 3-1. An application server also serves as a platform to develop and deploy web services and Enterprise JavaBeans (EJBs), and as a transaction and messaging engine when delivering business logic to users on a variety of client devices.

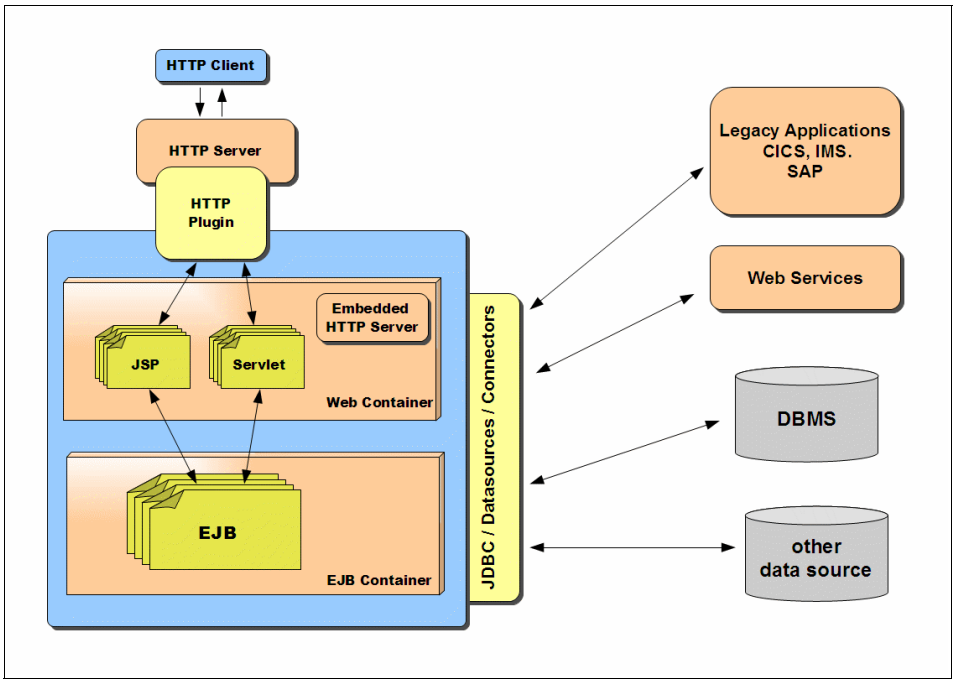


Figure 3-1 Basic architecture of an application server

The application server acts as middleware between back-end systems and clients. It provides a programming model, an infrastructure framework, and a set of standards for a consistent designed link between them.

WebSphere Application Server provides the environment to run your solutions and to integrate them with every platform and system as business application services conforming to the SOA reference architecture.

WebSphere Application Server is a key SOA building block. From a SOA perspective, with WebSphere Application Server you can perform the following functions:

- Build and deploy reusable application services quickly and easily
- Run services in a secure, scalable, highly available environment
- Connect software assets and extend their reach
- Manage applications effortlessly
- Grow as your needs evolve, reusing core skills and assets

Figure 3-2 shows the SOA reference architecture and WebSphere Application Server's role in this architecture.

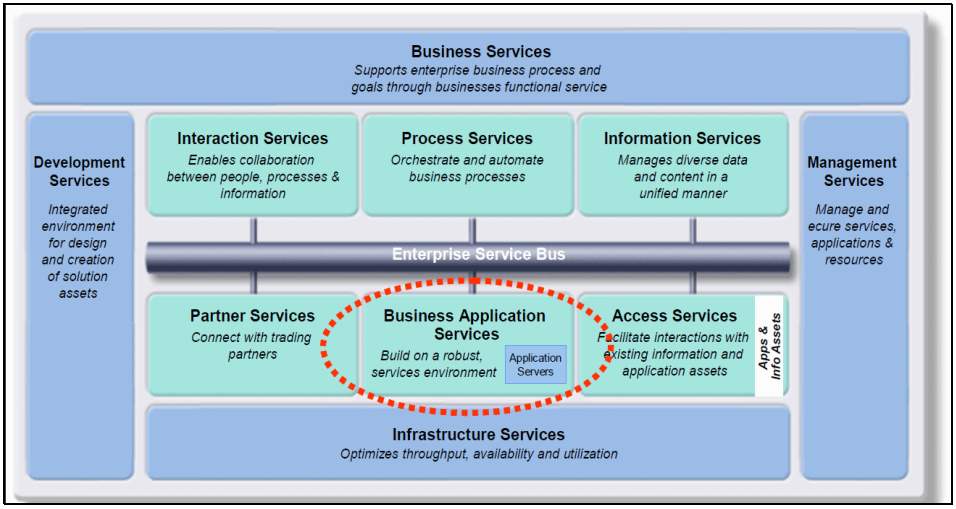


Figure 3-2 Business application services position in SOA reference architecture

WebSphere Application Server is available on a wide range of platforms and in multiple packages to meet specific business needs. By providing the application server that is required to run specific applications, it also serves as the base for other WebSphere products, such as IBM WebSphere Enterprise Service Bus, WebSphere Process Server, WebSphere Portal, and many other IBM software products.

As business needs evolve, new technology standards become available. The reverse is also true. Since 1998, WebSphere has grown and adapted itself to new technologies and to new standards to provide an innovative and cutting-edge environment that allows you to design fully-integrated solutions and to run your business applications.

3.2.2 Packaging

Because e-business application scenarios require various levels of application server capabilities, WebSphere Application Server is available in multiple packaging options. Although they share a common foundation, each option provides unique benefits to meet the needs of applications and the infrastructure that supports them. At least one WebSphere Application Server product fulfills the requirements of any particular project and its supporting infrastructure. As your business grows, the WebSphere Application Server family provides a migration path to more complex configurations.

The following packages are available:

- WebSphere Application Server - Express V7.0
- WebSphere Application Server V7.0
- WebSphere Application Server for Developers V7.0
- WebSphere Application Server Network Deployment V7.0
- WebSphere Application Server for z/OS V7.0

Figure 3-3 summarizes the main components included in each WebSphere Application Server package.

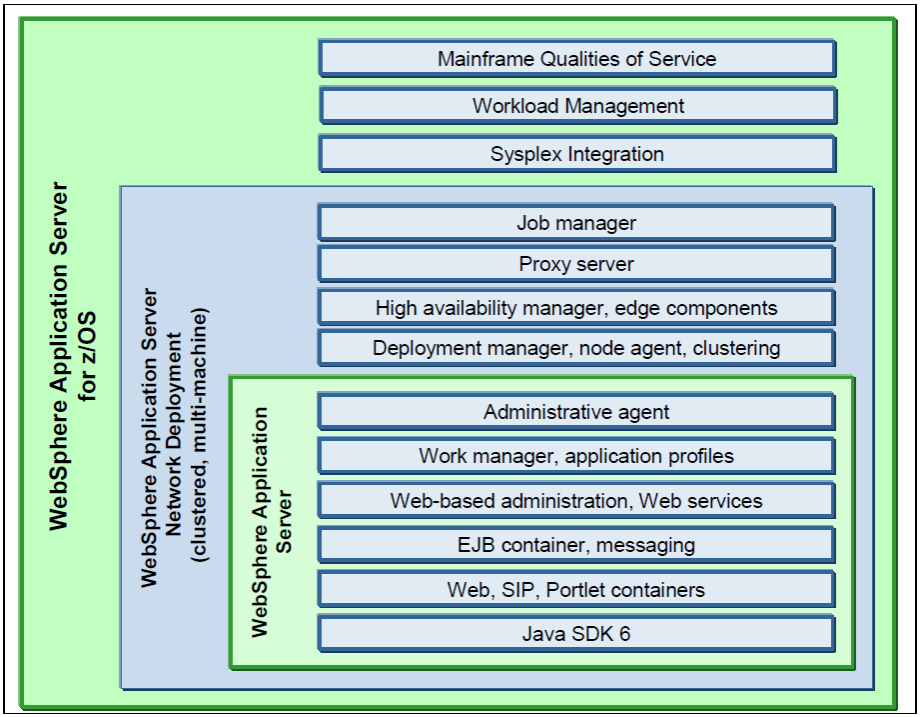


Figure 3-3 Packaging structure of WebSphere Application Server V7.0

Where to look for more information: For more information about WebSphere Application Server packaging, see *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708.

WebSphere Application Server: Express V7.0

The Express package is geared to those who need to get started quickly with a strong and affordable application server based on standards. It is specifically targeted at medium-sized businesses or departments of a large corporation, and is focused on providing ease of use and ease of application development. It contains full Java EE 5, EJB 3.0, and web services support, but is limited to a 32-bit single-server environment and to a maximum of 240 Processor Value Units (PVUs) per server for licensing purposes.

It also includes feature pack support. This package does not provide clustering and high availability features. For more information about WebSphere Application Server - Express V7.0, see the following web page:

`http://www.ibm.com/software/webservers/appserv/express/`

WebSphere Application Server V7.0

Also referred to as the base, this package is the next level of server infrastructure in the WebSphere Application Server family. Although the WebSphere Application Server package is functionally equivalent to Express (single-server environment), this package differs in packaging and licensing. It is available for both 32-bit and 64-bit platforms. This package is ideal for lightweight application solutions where cost and simplicity are key. For more information about WebSphere Application Server V7.0, see the following web page:

`http://www.ibm.com/software/webservers/appserv/was/`

WebSphere Application Server for Developers V7.0

The WebSphere Application Server for Developers package is functionally equivalent to the WebSphere Application Server package but it is licensed for development use only. WebSphere Application Server for Developers is an easy-to-use development environment to build and test applications for your SOA. For more information about WebSphere Application Server V7.0 for Developers, see the following web page:

`http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/`

WebSphere Application Server Network Deployment V7.0

WebSphere Application Server Network Deployment (ND) provides capabilities to develop more enhanced server infrastructures. It extends the WebSphere Application Server base package and includes the following features:

- Clustering capabilities
- Edge components
- Dynamic scalability
- High availability
- Advanced management features for distributed configurations

These features become more important at larger enterprises, where applications tend to service a larger client base, and more elaborate performance and high availability requirements are in place.

WebSphere Application Server Network Deployment Edge Components provide high performance and high availability features. For example, the Load Balancer (a software load balancer) provides horizontal scalability by dispatching HTTP requests among several web server or application server nodes supporting various dispatching options and algorithms to assure high availability in high volume environments. The usage of Edge Component Load Balancer can reduce web server congestion, increase content availability, and provide scaling ability for the web server.

WebSphere Application Server Network Deployment also includes a dynamic cache service, which improves performance by caching the output of servlets, commands, web services, and JSP files. This cache can be replicated to the other servers. The state of dynamic cache can be monitored with the cache monitor application. For more information about WebSphere Application Server Network Deployment V7.0, see the following web page:

http://www.ibm.com/software/webservers/appserv/was/network/

WebSphere Application Server for z/OS V7.0

IBM WebSphere Application Server for z/OS is a full-function version of WebSphere Application Server Network Deployment. It offers all the options and functions common to WebSphere Application Server V7.0 on distributed platforms.

WebSphere Application Server for z/OS V7.0 enhances the product in a variety of ways:

- Defines Service Level Agreements (SLA) on a transaction base (response time per transaction).
- Protects your production with Workload Management, in times of unpretendable peaks.
- Uses z/OS functionality for billing based on used resources or transactions.
- Uses one central security repository, including Java role-based security.
- Builds a cluster inside of a single application server (multi-servant).
- Profits from near linear hardware and software scalability.
- Profits from System z® cluster (Parallel Sysplex®) and up to 99.999% availability.

For more information about WebSphere Application Server for z/OS V7.0, see *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708, or visit the following web page:

`http://www.ibm.com/software/webservers/appserv/zos_os390/`

3.2.3 Java Platform Enterprise Edition (Java EE)

The technology that powers WebSphere products is Java. Over the years, many software vendors have collaborated on a set of server-side application programming technologies that help build web-accessible, distributed, platform-neutral applications. These technologies are collectively branded as the Java Platform, Enterprise Edition (Java EE). This contrasts with the Java Platform, Standard Edition (Java SE) platform, with which most clients are familiar. Java SE supported the development of client-side applications with rich graphical user interfaces (GUIs).

The Java EE platform, built on top of the Java SE platform, provides specifications for developing multi-tier enterprise applications with Java. It consists of application technologies for defining business logic and accessing enterprise resources such as databases, enterprise resource planning (ERP) systems, messaging systems, internal and external business services, email servers, and so forth.

Java Platform name changes: Java EE and Java SE were formerly named Java 2 Platform, Enterprise Edition (J2EE) and Java 2 Platform, Standard Edition (J2SE) respectively. After this name change, J2EE 1.5 is renamed as Java EE 5. J2SE 6 is renamed as Java SE 6. This book covers these standards with their new names.

The potential value of Java EE to clients is tremendous. This includes the following benefits:

- An architecture-driven approach allowing application development helps reduce maintenance costs and allows for construction of an information technology (IT) infrastructure that can grow to accommodate new services.
- Application development standards, tools and predefined rules improve productivity and accelerates and shortens development cycles.
- Packaging, deployment, and management standards for your enterprise applications facilitate systems and operations management
- Industry standard technologies enable clients to choose among platforms, development tools, and middleware to power their applications.
- Platform independence gives flexibility to create an application one time and to run it on multiple platforms. This provides true portability to enterprise applications.
- Embedded support for Internet and web technologies allows for applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Java EE 5, the latest release of the Java EE platform, represents a significant evolution in the Java enterprise programming model. It improves the application developer experience and productivity with the following new features:

- Latest specifications
  - Enterprise JavaBeans (EJB) 3.0
  - JAX-WS 2.0
  - JSP 2.1
  - Servlet 2.5
  - JSF 1.2
- Java Development Kit (JDK) 6.0
- Usage of progressive disclosure
- Annotations and injection support to reduce complexity
- EJB development as Plain Old Java Objects (POJOs)
- Java Persistence API (JPA) to allow creation of simpler entities using annotated POJO model

Another exciting opportunity for IT is web services. Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols (such as HTTP and XML) already in use. This allows for easy integration of both intra- and inter-business applications, which can lead to increased productivity, reduced expense, and shortened time to market. Web services are also the key elements of SOA, which provides for the reuse of existing service components and more flexibility to enable businesses to address changing opportunities.

3.2.4 Evolving Java application development standards

WebSphere Application Server V7.0 provides the runtime environment for applications that conform to the J2EE 1.2, 1.3, 1.4, and Java EE 5 (formerly J2EE 1.5) specifications.

Java SE V6 support adds the ability to invoke the Java compiler from the Java Virtual Machine (JVM). It ncludes scripts with the ability to access APIs in the JVM.

Feature Pack for EJB 3.0: The Feature Pack for EJB 3.0 available for WebSphere Application Server V6.1 is embedded into WebSphere Application Server V7.0.

If you have used EJB 3.0 Feature Pack functionality in your application in a WebSphere Application Server V6.1 environment before, there is no need to install additional WebSphere Application Server packages to use these applications with V7.0.

Refer to 3.2.7, “Feature packs” on page 55 to learn more about feature packs for WebSphere Application Server V7.

For universal data access and persistence, WebSphere Application Server supports the following specifications:

- Java DataBase Connectivity (JDBC) API 4.0

  You can connect to any type of data sources with the JDBC API.

- Java Persistence API (JPA)

  JPA delivers simplified programming models and a standard persistence API for building reusable persistent objects.

- Service Data Objects (SDO)

  With SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources as collections of tree-structured or graph-structured objects. WebSphere Application Server V7.0 extends the application server to support the following tasks:

  – Java Specification Requests (JSR) 286 (Portlet 2.0) compliant portlets

  – Session Initiation Protocol (SIP) applications conforming to the JSR 116 specification

  – Java Servlet 2.5 (JSR 154) and JavaServer Pages (JSR 245) specifications for web applications

WebSphere Application Server’s enhanced support for application development standards delivers maximum flexibility and significantly improves developer productivity.

3.2.5 Supported hardware, platforms, and software

This section details the hardware, platforms, and software versions that WebSphere Application Server V7.0 supports at the time this book was written.

For the most up-to-date operating system levels and hardware requirements, refer to the WebSphere Application Server system requirements, at the following web page:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

Hardware

Supported hardware for WebSphere Application Server V7.0:

- IBM POWER® family of processors
- PA-RISC processor
- Intel® Itanium® 2 processor
- Intel Pentium® processor at 500 MHz or faster
- Intel EM64T or AMD Opteron
- IBM System z processors
- IBM System i® (iSeries)
- SPARC workstations

Operating systems

Table 3-1 shows supported operating systems and their versions for WebSphere Application Server V7.0.

Table 3-1 Supported operating systems for WebSphere Application Server V7.0

Operating system	Supported Versions
Microsoft Windows	Supported with 32-bit WebSphere Application Server: ► Microsoft Vista Business ► Microsoft Vista Enterprise ► Microsoft Windows Server 2003 with SP2 ► Microsoft Windows Server 2003 R2 ► Microsoft Windows Server 2008, Datacenter with SP1 ► Microsoft Windows Server 2008, Enterprise with SP1 ► Microsoft Windows Server 2008, Standard with SP1 ► Microsoft Windows XP Professional with SP2 Supported with 64-bit WebSphere Application Server: ► Microsoft Windows Server 2003 x64 R2 ► Microsoft Windows Server 2008 Enterprise x64 with SP1 ► Microsoft Windows Server 2008 Datacenter x64 with SP1 ► Microsoft Windows Server 2008 Standard x64 with SP1
IBM AIX	Supported with 32-bit and 64-bit WebSphere Application Server: ► AIX 6.1 with Maintenance Package 6100-00-04 ► AIX 5L™ 5.3 with Service Pack 5300-07-01
Sun Solaris on Sparc	Supported with 32-bit WebSphere Application Server: ► Sun Solaris Version 9 with Patch Cluster dated 1/03/08 or later ► Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later Supported with 64-bit WebSphere Application Server: ► Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later
Sun Solaris on x64	Supported with 64-bit WebSphere Application Server: ► Sun Solaris Version 10 with Patch Cluster dated 1/07/08 or later
HP-UX on PA-RISC	Supported with 32-bit WebSphere Application Server: ► HP-UX 11iv2 with Patch Bundle dated Dec 2007 ► HP-UX 11iv3 with Patch Bundle dated Sep 2007
HP-UX on Itanium	Supported with 64-bit WebSphere Application Server: ► HP-UX 11iv2 with Patch Bundle dated Dec 2007 ► HP-UX 11iv3 with Patch Bundle dated Sep 2007
Linux on x86	Supported with 32-bit WebSphere Application Server: ► Red Hat Enterprise Linux AS, Version 4 with Update 6 ► Red Hat Enterprise Linux ES, Version 4 with Update 6 ► Red Hat Enterprise Linux WS, Version 4 with Update 6 ► Red Hat Enterprise Linux, Version 5 with Update 1 ► SUSE Linux Enterprise Server, Version 9 with SP4 ► SUSE Linux Enterprise Server, Version 10 with Update 1

Operating system	Supported Versions
Linux on x86-64 and POWER	Supported with 32-bit and 64-bit WebSphere Application Server: ► Red Hat Enterprise Linux, Version 4 with Update 6 ► Red Hat Enterprise Linux, Version 5 with Update 1 ► SUSE Linux Enterprise Server, Version 9 with SP4 ► SUSE Linux Enterprise Server, Version 10 with Update 1
Linux on System z and zSeries®	Supported with 31-bit and 64-bit WebSphere Application Server: ► Red Hat Enterprise Linux AS, Version 4 with Update 6 ► Red Hat Enterprise Linux, Version 5 with Update 1 ► SUSE Linux Enterprise Server, Version 9 with SP4 ► SUSE Linux Enterprise Server, Version 10 with Update 1
IBM z/OS (Supported for WebSphere Application Server V7.0 for z/OS)	Supported with 31-bit and 64-bit WebSphere Application Server: ► z/OS V1.7 ► z/OS V1.8 ► z/OS V1.9 ► z/OS.e ► z/OS.e V1.7 ► z/OS.e V1.8
IBM i	► IBM i V5R4 ► IBM i V6R1

About support limitations: To see if there is a 32-bit or 64-bit support limitation for your operating system, refer to the WebSphere Application Server system requirements web page:

http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html

Web servers

Supported web servers for WebSphere Application Server V7.0 are as follows:

- Apache HTTP Server 2.0.58
- Apache HTTP Server 2.2
- IBM HTTP Server for WebSphere Application Server V6.1
- IBM HTTP Server for WebSphere Application Server V7.0
- Internet Information Services (IIS) 6.0
- Internet Information Services (IIS) 7.0
- Lotus® Domino® Enterprise Server 7.0.2
- Lotus Domino Enterprise Server 8.0
- Sun Java System Web Server 6.1 SP8
- Sun Java System Web Server 7.0 Update 1

Important: Not every web server is supported on all platforms and operating systems. To match compatibility and support, see the original documentation for each of these software vendors.

Database servers

Table 3-2 shows the database servers that WebSphere Application Server V7.0 supports.

Table 3-2 List of databases that are supported by WebSphere App. Server V7

Databases	Supported versions
IBM DB2 [a]	► DB2 for iSeries 5.3, 5.4, or 6.1 ► DB2 for z/OS V8, V9 or V9.1 ► DB2 Connect™ ► DB2 Enterprise Server Edition V8.1, V8.2, V9.0 and V9.5 ► DB2 Express V8.1, V8.2, V9.0 and V9.5 ► DB2 Workgroup Server Edition V8.1, V8.2, V9.0 and V9.5
Cloudscape	► Cloudscape 10.3 (Derby)
Oracle	► Oracle 10g Standard/Enterprise Release 1 - 10.1.0.5 ► Oracle 10g Standard/Enterprise Release 2 - 10.2.0.3 ► Oracle 11g Standard/Enterprise Release 1 - 11.1.0.6
Sybase	► Sybase Adaptive Server Enterprise 12.5.4 ► Sybase Adaptive Server Enterprise 15.0.2
Microsoft SQL Server	► Microsoft SQL Server Enterprise 2005 SP2
Informix®	► Dynamic Server 10.00xC6 ► Dynamic Server 11.10xC1 ► Dynamic Server 11.5xC1
IMS™	► IMS V9 ► IMS V10
WebSphere Information Integrator	► WebSphere Information Integrator 8.2 FP8 ► WebSphere Information Integrator 9.1 FP3 ► WebSphere Information Integrator 9.5 FP2

a. For a detailed list of supported fixpack levels and editions for DB2, see the support document #7013265 at the following web page:
http://www.ibm.com/support/docview.wss?rs=180&uid=swg27013265

Important: Not every database server is supported on all platforms and operating systems. To match compatibility and support, see the original documentation for each of these software vendors.

Directory servers

Supported directory servers for WebSphere Application Server V7.0 are as follows:

- LDAP Servers using Stand Alone LDAP User Registry Configuration and Federated Repository Configuration:
  - IBM Tivoli Directory Server 6.0
  - IBM Tivoli Directory Server 6.1
  - IBM z/OS Integrated Security Services 1.8
  - IBM z/OS Integrated Security Services 1.9
  - IBM z/OS.e Security Server 1.8
  - IBM z/OS.e Security Server 1.9
  - Lotus Domino Enterprise Server 7.0
  - Lotus Domino Enterprise Server 8.0
  - Novell eDirectory 8.7.3 SP9
  - Novell eDirectory 8.8.1
  - Sun Java Directory Server 6.0
  - Sun Java Directory Server 6.1
  - Windows Active Directory 2003
  - Windows Active Directory 2008
- LDAP Servers using only Federated Repository Configuration:
  - Windows Active Directory Applications Mode 1.0 SP1
  - OpenLDAP 2.4.7

3.2.6 New features and enhancements in WebSphere Application Server V7

WebSphere Application Server V7 builds upon the powerful and stable core of previous releases with several new features and enhancements. In addition to support for the latest standards and programming models, V7 contains key improvements in the areas of systems management, installation, and security. Together, these features further expand on WebSphere Application Server's platform coverage, runtime management capabilities, and application deployment options to decrease costs and grow businesses.

This section offers a look at the key features that enable this new release to provide an even more flexible and reliable foundation for your SOA environment.

Where to look for more information: For more information about what is new in WebSphere Application Server V7, see the developersWorks article *What's new in WebSphere Application Server V7?* available on the following web page:

http://www.ibm.com/developerworks/websphere/library/techarticles/0809_alcott/0809_alcott.html.

Excerpt: The rest of this chapter, from page 44 to page 68 is an excerpt from the developersWorks article *What's new in WebSphere Application Server V7?*, mentioned in the previous note.

Standards

WebSphere Application Server V7 now includes new support for the following functions:

Java EE 5

The most notable supported standard in WebSphere Application Server V7 is Java Platform, Enterprise Edition (Java EE) 5. WebSphere Application Server V7 provides complete support for the Java EE 5 specification, including the web services and EJB 3.0 functions that were previously available as feature packs in V6.1.

In case you are not familiar with Java EE 5, this latest release of the standard represents a significant evolution in the Java enterprise programming model, resulting in considerable improvements in the application developer experience and, in turn, in application developer productivity. A phrase often used to describe the Java EE 5 programming model is *progressive disclosure*, meaning that much of the code that has, until now, been required for Java EE development is eliminated. Instead, the most generally used application context is provided as the default behavior. Then, using annotations, you can override the defaults, if required, to achieve the required implementation. In this manner, the application is progressively constructed, and only to the extent necessary.

WebSphere Application Server V7 also introduces support for Java Platform, Standard Edition (Java SE) 6.

Dependency injection

Developer productivity is further enhanced when it is time to override the defaults, because this can be accomplished using annotations rather than writing code. Annotations are used in conjunction with a programming pattern known as *dependency injection*, or *Inversion of Control* (IoC), in which the application code

declares variables and annotates them to indicate whatever is needed, and then the container inserts the specified object or resource references.

Table 3-3 shows a simple example to help explain dependency injection. The code segment on the left is from an EJB 2.1 application. The one on the right, from an EJB 3.0 application, has an annotation @EJB that is used to tell the container that this is an EJB. The container then inserts the EJB 3.0 application with the required wrapper (or boilerplate), eliminating the need for you to do so.

Table 3-3 Dependency Injection

EJB 2.1	EJB 3.0

`Object obj = Context.Lookup("java:comp/env/ejb/MyCartHome");`
`CartHome theCartHome = (CartHome)`
`PortableRemoteObject.narrow(obj,CartHome.class);`
`ShoppingCart myCart = theCarthome.create()`
`myCart.someShoppingMethod();`
	`@EJB`
	`ShoppingCart myCart`
	`myCart.someshoppingMetthod`

Java Persistence API

Another aspect to the simplification of the programming model in Java EE 5 is the ability to develop EJB components as Plain Old Java Objects (POJOs). Simplifying EJB development even further is the Java Persistence API (JPA), which enables entities to be created using annotated POJOs. Aside from making EJB development and use even easier, this also enables Java SE developers to quickly learn Java EE and develop enterprise applications.

Web services

The improvements to the Java EE 5 programming model are not limited to EJB development. Web services development is also greatly simplified through annotations. In Table 3-4, a Java EE 1.4 JAX-RPC application is depicted on the left and the corresponding Java EE 5 JAX-WS application is depicted on the right.

Table 3-4 Web services development

JAX-RPC 1.1 code	JAX-WS 2.0 code

`public interface StockQuote extends Remote{`
`  public float getQuote(String sym) throws`
`  Remote Expecption;`
`}`
`public class QuoteBean implements {`
`  public float getQuote(String sym) {...}`
`}`
	`@WebService public interface StockQuote {`
	`  public float getQuote(String sym);`
	`}`
	`@WebService public class QuoteBean`
	`implements Stockquote {`
	`  public float getQuote(String sym);`
	`}`

The JAX-WS implementation in WebSphere Application Server V7 also provides support for SOAP 1.2, and for generic XML or HTTP as a protocol binding, which enables the creation of WS clients and WS providers that do not use SOAP for their wire level message format.

Another important addition in WebSphere Application Server V7 is its support for Message Transmission Optimization Mechanism (MTOM), an improved method for sending binary attachments. MTOM is the preferred attachment mechanism for non-Java EE-based web services, so support for MTOM improves the interoperability between Java EE web services and non-Java EE web services.

Portlet 2.0 API

WebSphere Application Server V7 also introduces support for the Portlet 2.0 API, also known as the Portlet JSR 286 specification.

Systems management

Systems management is another major area enhanced in WebSphere Application Server V7, with a number of new management and administration options to help you implement a more flexible, scalable, and asynchronous administrative topology. The following sections provide more detail.

Intelligent provisioning

Intelligent provisioning by the WebSphere Application Server run time selects only the run time function needed for an application, reducing both memory footprint and application server startup time. Each application is examined by WebSphere Application Server at application installation time to generate an activation plan. At run time, the server starts only those components that are required by the activation plan.

WebSphere Application Server V7 contains support for a Web/JDBC application scenario in which only the web container and core run time components are started. This is in contrast to V6.1, in which the EJB container, SIP container, and web services run time also has been started, as shown in Figure 3-4 on page 47. This capability is also used in fixed function servers (for example, WebSphere Application Server proxy server) to help reduce the memory footprint, and is designed for extension by other WebSphere products that run on top of WebSphere Application Server.

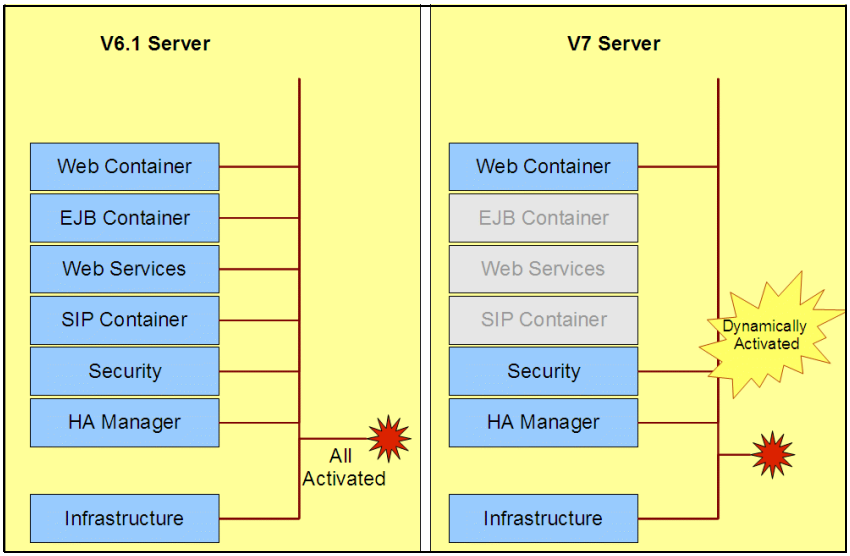


Figure 3-4 Web services run time started

Additional run time footprint reductions are delivered in the form of the Java SE 6 implementation that ships with WebSphere Application Server V7. The shared class cache that was introduced in the IBM Java SE 5 implementation has been improved so that the cache can now be persistent. As a result, the cache can survive a shutdown and restart of all WebSphere processes on a server. In the case of an application developer running a single WebSphere Application Server instance, the contents of the shared class cache can be reloaded on application server startup. When used independently (or in combination with intelligent provisioning) the result is significantly faster application server startup times.

Another improvement in the IBM Java SE 6 is the use of compressed references (or pointer compression) in 64-bit WebSphere Application Server JVMs. The use of compressed references can provide a significant reduction in the process footprint of a 64-bit JVM, when compared to a 32-bit JVM. Prior to the IBM Java SE 6 implementation, it was not unusual for a 64-bit heap to be 1.7 to 2 times the size of an equivalent 32-bit heap.

Administrative agent

Flexible management is an improvement that can help reduce the cost of administering large deployments in several ways. The easiest way to illustrate this new feature is by comparing the WebSphere Application Server (base) V6.1 run time model to what is provided by flexible management in V7. In V6.1, the administrative run time is hosted in WebSphere Application Server. This means that the administrative console runs in an application server. When wsadmin

scripts are run against an application server, each application server process has administrative overhead as a result. There is also no provision for managing multiple base application servers profiles. Each must be managed independently, as depicted in Figure 3-5.

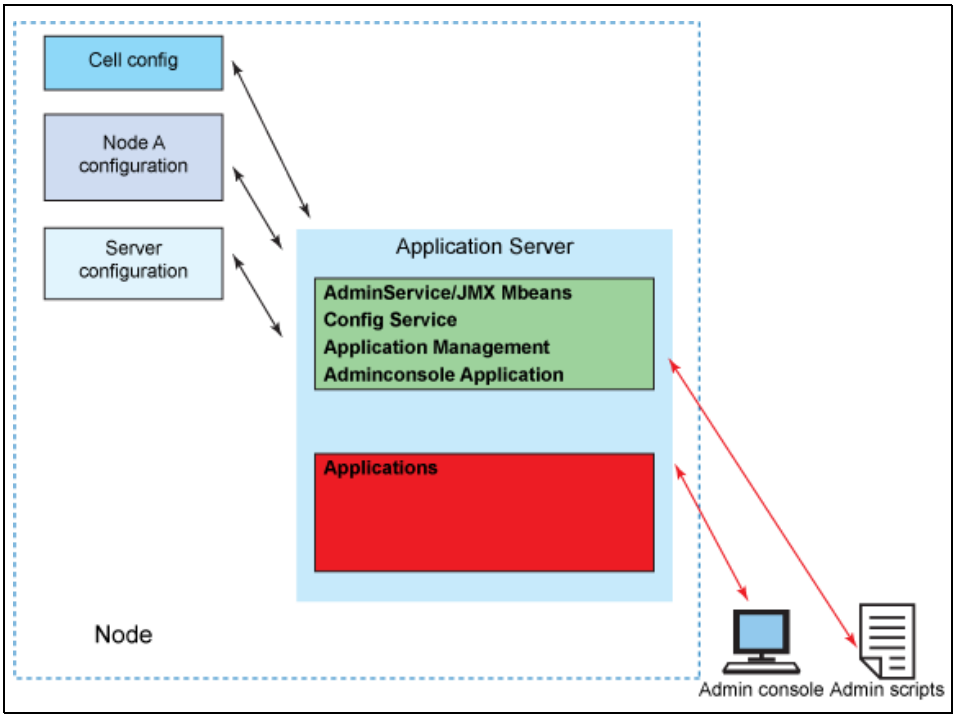


Figure 3-5 Multiple base profiles

Flexible management changes this picture by separating most of the administrative components from the application server run time. Now, each application server is no longer a single point of management. This is accomplished with a separate administration process for (base) WebSphere Application Servers known as the *administrative agent*, shown in Figure 3-6.

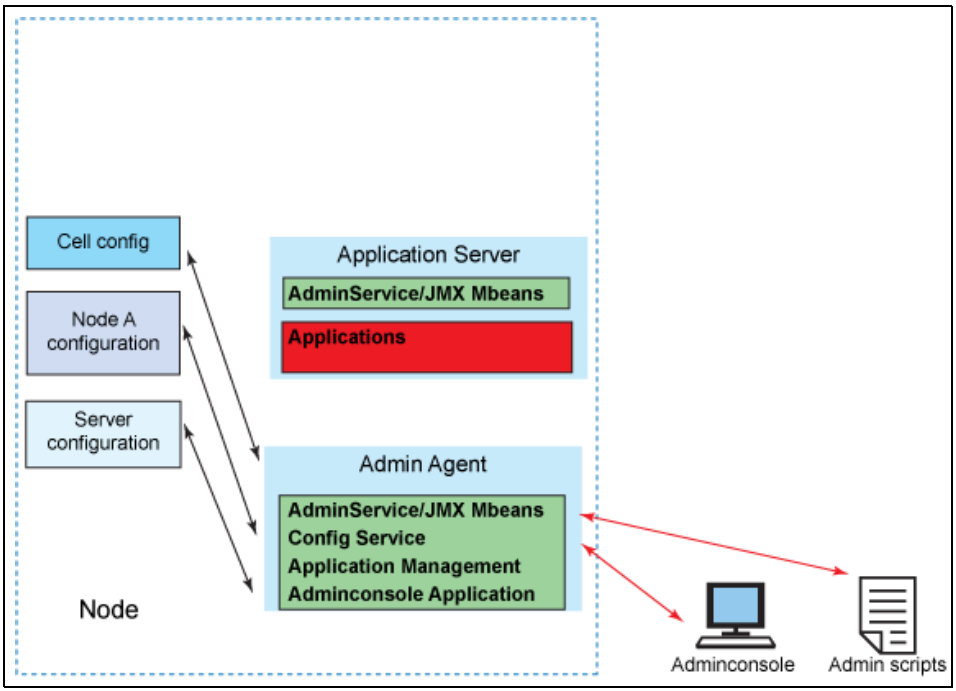


Figure 3-6 Administrative agent

You might wonder if moving the administration function from the application server to the administrative agent provides more capability. The answer lies in the fact that an administrative agent can manage multiple WebSphere Application Server (base) instances on a single machine, as shown in Figure 3-7 on page 50. When using an administrative agent, as the number of application server instances increases, the duplication of the administration footprint (for each application server) is eliminated. Further, because the administrative agent can manage all the application server instances on a machine, there is only one set of remote administration ports to remember (those for the administrative agent), as opposed to trying to manage all the separate administration ports associated with each application server.

When working with the admin agent, remember the following factors:

- The administrative agent can only manage the application servers that are on the same machine as the administrative agent.
- Although the administrative agent can manage multiple application servers, each application server is managed individually. In other words, log into the administrative agent to change the configuration of server1, log out, then log on to server2, and so on.
- The administrative agent only provides management of these application servers (and their applications). It does not provide for clustering and failover. For clustering and failover and centralized application management, you still need WebSphere Application Server Network Deployment.

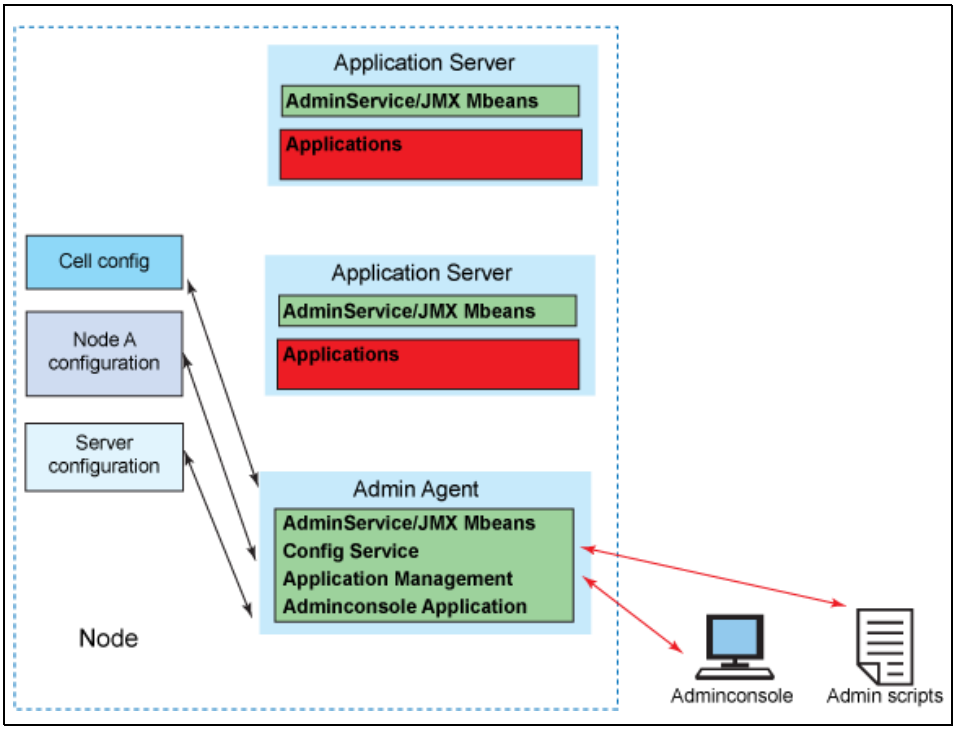


Figure 3-7 Administrative agent managing multiple application servers

Job manager

The job manager is another aspect of flexible management that is available in WebSphere Application Server Network Deployment V7. As with the administrative agent, it might be best to describe how the job manager functions by quickly reviewing the run time model for V6.1.

In WebSphere Application Server Network Deployment V7, there is a tight coupling:

- Between application servers and node agents.
- Between node agents and the deployment manager.

This tight coupling can impact the scalability of the administrative run time if the run time components are not located together in close proximity. A possible example might be a branch office deployment, where a deployment manager in a central location is being used to manage nodes dispersed in each remote branch office. Even under the best of conditions (redundant, high capacity, low latency networks between the central location and the branch offices, and so on) this can be problematic because Wide Area Networks (WANs) are rarely as reliable as Local Area Networks (LANs), and both are known to occasionally fail.

The job manager in WebSphere Application Server Network Deployment V7 addresses the limitations inherent in the current management architecture by adopting a loose management coupling architecture. Rather than synchronously controlling a number of remote endpoints (node agents), the job manager coordinates management across a group of endpoints by providing an asynchronous job management capability across a number of nodes. The management model relies on the submission of management jobs to these remote endpoints, which can be either a WebSphere Application Server (base) administrative agent or WebSphere Application Server Network Deployment V7 deployment manager. In turn, the administrative agent or the deployment manager is what actually executes the jobs that update the configuration, start or stop applications, or perform a variety of other common administrative tasks.

The management model provided by the job manager is shown in Figure 3-8 on page 52. It is important to understand that the job manager is not a replacement for a deployment manager. It is an option for remotely managing a WebSphere Application Server Network Deployment V7 deployment manager or, more likely, multiple deployment managers. Recall the aforementioned example, trying to manage multiple remote nodes from a single deployment manager. With the job manager, each branch can be converted into its own WebSphere Application Server Network Deployment V7 cell with a deployment manager running locally in each branch. The job manager can be used to remotely manage each of these remote cells.

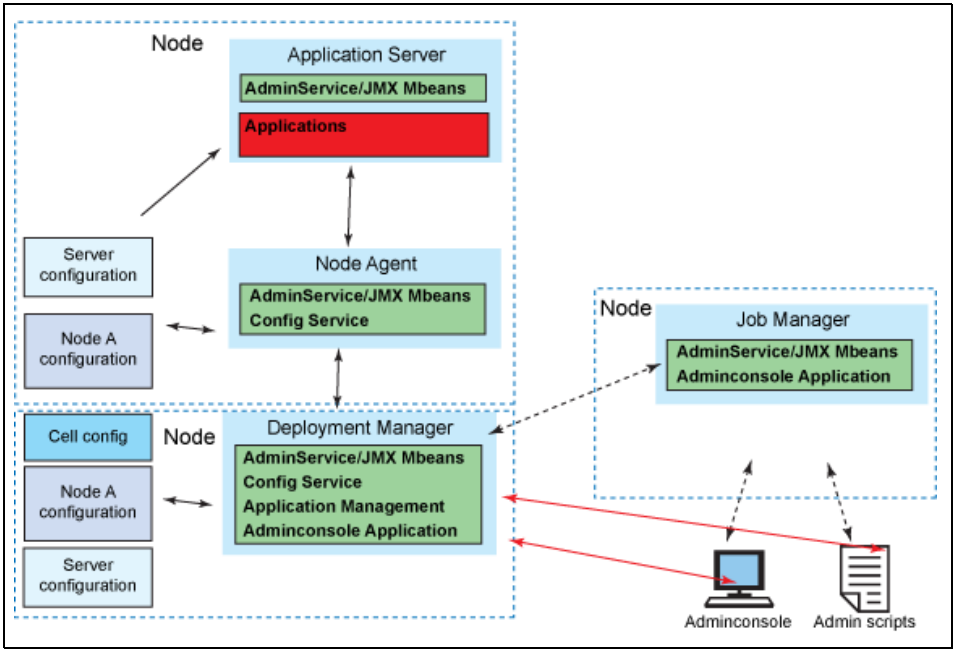


Figure 3-8 Job manager management model

Web services

Another important new systems management capability in V7 is in the area of web services. Web service policy sets are mechanisms provided by WebSphere Application Server V7 for centrally defining various quality of service policies to apply to web services that are already deployed. There are two types of policy sets:

- Application policy sets

  These policy sets are used for business-related assertions, such as business operations that are defined in the WSDL file.

- System policy sets

  These policy sets are used for non-business-related system messages, such as messages that are defined in specifications that apply QoS policies. Examples include security token (RST) messages that are defined in WS-Trust, or the creation of sequence messages that are defined in WS-Reliable Messaging metadata.

Rather than defining individual policies and applying them individually to each web service, policy sets can be applied one time to all web service applications to which they are applicable, thus ensuring a uniform quality of service for a given type of web service. Related to this, WebSphere Service Registry and

Repository discovers WebSphere Application Server V7 JAX-WS policy sets, discovers existing associations, and represent those as policy attachments. A number of default policy sets are provided:

- LTPA WSSecurity Default
- Kerberos V5 HTTPS default
- SSL WSTransaction
- Username SecureConversation
- Username WSSecurity default
- WS-Addressing default
- WSHTTPS default
- WS-I RSP ND
- WS-ReliableMessaging persistent

Additionally, policy sets can be customized or modified as required.

Messaging

Beyond the web services improvements in WebSphere Application Server V7, there are also improvements to a related SOA component: messaging. New administrative wizards for System Integration Bus (SIBus) configuration are provided for configuring a cluster bus member and a foreign bus connection, and new panels for SIBus authorizations, WebSphere MQ JCA Resource Adapter configuration, and for inspecting an application's use of SIBus resources.

V7 also includes a WebSphere MQ JMS JCA 1.5 resource adapter, and associated new panels and administrative commands. Having the JCA 1.5 resource adapter also means that message-driven beans that consume messages from WebSphere MQ can now use activation specifications instead of listener ports.

Improved management of activation specifications (stop, start, message retry limits [or “poison” message filtering]) is also included in WebSphere Application Server V7.

Centralized installation

WebSphere Application Server Network Deployment V7 also adds the capability to perform centralized installation from the deployment manager to remote nodes. This enables a single installation to the deployment manager to be pushed out as an installation package from the deployment manager to a series of endpoints. The endpoint can be either a node that is not part of a WebSphere Application Server Network Deployment V7 cell (in which case Network Deployment can be installed) or an existing WebSphere Application Server Network Deployment V7 node that might need a fix pack.

Business level applications

Another new function in WebSphere Application Server V7 is a the notion of an application beyond J2EE, or a *business level application* (BLA). This is a new administration concept that expands upon what was previously provided by J2EE. Cases where this can be of value include those where an application has the following characteristics:

- Is composed of multiple packages
- Contains additional libraries, or non-Java EE artifacts
- Includes artifacts that run on heterogeneous (WebSphere and non-WebSphere) run times
- Is defined in a recursive manner. For example, if an application includes other applications

Essentially, a BLA is a grouping notion for managing Java EE and other artifacts under a single application definition. A BLA can span more than WebSphere Application Server deployment target run times, such as a proxy server, web server, WebSphere Application Server Community Edition, and so on. In addition, a BLA provides full life cycle management of applications: install, distribute, activate, monitor, update, and remove.

Security

Security management improvements are also a significant portion of new features in WebSphere Application Server V7. Perhaps the most prominent change is the ability to create multiple security domains within a single WebSphere Application Server cell. Each security domain can have its own user population (and underlying repository), and, additionally, the application domain can be separated from the administrative domain. Not only can each domain have its own separate user population, but each domain can also be customized with a separate security configuration (for example, JAAS login configuration, TAI, authorization provider, JCA authentication data) in addition to the current application server level security configuration options. The scope of the domain is variable and can range from a cell, cluster, or node to a node group or application server. As a result, applications have a distinct user population and distinct security configuration.

Further, the fine grained administrative security function introduced in WebSphere Application Server V6.1 but limited to wsadmin is now expanded in V7 to include the administrative console. Within the administrative security domain for a cell, this enables administrative roles to be restricted to specific components, such as a cluster, node, application server, or application.

Kerberos

WebSphere Application Server V7 provides support for Kerberos authentication, which can be used in a variety of single sign-on scenarios. The Kerberos support in V7 expands on the SPNEGO TAI support in V6.1 to provide better interoperability and identity propagation with other applications that support Kerberos authentication (such as IBM DB2, .NET, and others).

3.2.7 Feature packs

A WebSphere Application Server feature pack is an optionally installable product extension for WebSphere Application Server that provides a set of new related standards and innovative features. With feature packs, users can take advantage of these new standards and features without having to wait for a new release of WebSphere Application Server.

Feature packs aim for simplifying the way you consume WebSphere Application Server by balancing customers' desire for less frequent releases and still making available the latest standards to customers who need them. With feature packs, customers can selectively take advantage of new standards and features and maintaining a more stable internal release cycle. IBM offers feature packs generally available or available in either open alpha, beta, or technology preview.

Where to look for more information: For more detailed information about the IBM WebSphere Application Server V7 feature packs, see the IBM Redbooks publication *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708, or visit the following web page:

`http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/`

Available WebSphere Application Server Feature Packs

The currently available feature packs for WebSphere Application Server V7.0 are detailed in the following sections.

IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0 Open Beta

The feature pack for OSGi Applications and JPA 2.0 provides a lightweight, simplified application framework to increase developer productivity and time to value. Through this feature pack, organizations can realize many of the benefits found in other nonstandard open source frameworks in a standardized, WebSphere integrated fashion.

This feature pack delivers open community- and standards-based implementations of the OSGi Blueprint service specification and Java EE 6 JPA 2.0 along with the ability, optionally, to assemble, deploy and manage applications as a collection of versioned OSGi bundles.

IBM WebSphere Application Server Feature Pack for XML

Since being standardized in 1999, XML use in application development environments has grown to include many scenarios:

- A simplified programming mode for building presentation flexible web applications using end-to-end declarative XML. XML presentation can easily be adapted to various platform types from mobile to desktop
- Access, search, retrieval, and presentation of data or documents stored in native XML
- Filter for XML data sets, such as web feeds in Java applications
- XML data transformations for application development scenarios

WebSphere Application Server is a leading platform for the latest application development standards that matter, including XML. The WebSphere Application Server V7 Feature Pack for XML delivers support for important XML standards to benefit various XML application development scenarios.

The WebSphere Application Server V7 Feature Pack for XML provides application developers with support for the Worldwide Web Consortium (W3C) XML standards XSLT 2.0, XPath 2.0, and XQuery 1.0. These new and updated W3C XML standards enable application developers with numerous advanced capabilities for building XML applications.

Highlights of the capabilities of the Feature Pack for XML are as follows:

- Simplifies XML application development and improve developer productivity
- Improves XML application performance through new features introduced in the W3C specifications to address previous shortcomings
- Improves performance, fidelity and ease of use by natively working with XML data when using standards-based skillsets
- Improves XML application reliability with new support for XML schema awareness and validation in XSLT 2.0 and XPath 2.0
- Enables developers to query large amounts of data stored in XML outside of a database with XQuery 1.0
- Supports Java applications that connect to WebSphere Application Server through an XML thin client

- Provides enterprise class multi-threaded scalability, serviceability with IBM support
- Offers consistent execution (invocation) and data navigation API, at the same time allowing access to existing Java logic across the XPath 2.0, XSLT 2.0, and XQuery 1.0
- Provides 40 samples of the new standards along with four end to end samples that show these standards in typical web applications

IBM WebSphere Application Server Feature Pack for Communications Enabled Applications (CEA)

Efficiently update existing applications with communication capabilities and reduce cost by leveraging existing Java skills.

The IBM WebSphere Application Server V7 Feature Pack for CEA delivers an SOA- and Java-based programming model for efficiently creating innovative multimodal communications applications,

Capabilities of the WebSphere Application Server V7 Feature Pack for CEA are as follows:

- Telephony access services
  - REST service and web service access to telephony services so your developers can use existing skills, without requiring communications expertise:
    - Make a call
    - Disconnect a call
    - Receive incoming call notifications
  - Simple telephony provisioning since CEA services are not in the call flow, but are asynchronously notified of a call by the customer's enterprise Telephony/Unified Communications solution using standard telephony SIP CTI Gateway (CSTA XML over SIP protocol, also known as ECMA TR/87)
- Multi-modal communications
  - Fully customizable JavaScript-based web 2.0 widgets to improve integration with your application and business process
  - Fully extensible JavaScript-based web 2.0 widgets using Dojo semantics to efficiently build new CEA capabilities for use across your applications and company

- Web 2.0 widget access to telephony services which can be easily and efficiently inserted into new or existing applications
  - Click to call

    Allows users to click a button and be connected to another user through telephone
  - Call Notifications

    Allows the application to poll for incoming calls which can then be programmatically answered to create a call session
- Web 2.0 widget access to multi-modal communications to efficiently build innovative applications that significantly improve user experiences and reduce user interaction costs
  - Collaboration Dialog

    A modal window that enables peer-to-peer data sharing through a REST service in a simple and extensible fashion
  - Contact Center Cobrowsing

    Enables a user and contact center representative to begin a “Click to call” telephone session. The user can then securely and privately browse a web application as WebSphere Application Server keeps their browser windows synchronized, with the ability to share navigation control, through Follow Me mode, or through individual web address sharing, and to highlight information for the other user to see.
  - Two-way synchronized forms

    An extension to the Collaboration Dialog web widget that allows two parties to input and visually confirm form data entered by the other party. Individual form fields, such as credit card number or social security number, can be masked to only display a subset of the field data entered to the reviewing party.
  - Peer-to-peer Cobrowsing

    Enables two users to begin a cobrowsing session through a unique web address. Users can then securely and privately browse a web application as WebSphere Application Server keeps their browser windows synchronized, with the ability to share navigation control, through Follow Me mode, or through individual web address sharing, and to highlight information for the other user to see.

- Enhanced security and privacy
  - Each user session is with and managed by WebSphere Application Server, not with each other. This provides a higher degree of security between users than a peer-to-peer session.
  - No client-side software installation is required for users, which reduces security-driven barriers to customers taking advantage of innovative user interaction features of your communications-enabled applications.
  - Using enterprise Telephony/Unified Communications network increases data privacy by keeping customer and company data on company infrastructure versus exposing the data to a third-party hosted solution.
- Telephony interoperability

  Tested with leading Telephony/Unified Communications solutions as an on-ramp to communications enabled business process (CEBP) solutions:
  - Avaya Aura Communication Manager
  - Cisco Unified Communications Manager
- Developer productivity
  - Sample reference application to speed developer learning
  - Unit test environment to prototype and test applications without requiring approval to access the corporate Telephony/Unified Communications network
  - Ability to use a third-party IP softphone within the provided unit test environment
- Developer tooling
  - Separately available in IBM Rational Application Developer V7.5, developers can build, test, and deploy communications enabled applications for WebSphere Application Server V7 with Rational Application Developer V7.5 and later.
- Function for SIP application developers
  - Support for SIP Servlet 1.1 (JSR 289)
  - Asynchronous Invocation API for accessing remote SIP sessions

IBM WebSphere Application Server V7 Feature Pack for Service Component Architecture

The WebSphere Application Server Feature Pack for Service Component Architecture (SCA) V1.0.1 helps deliver simplified development by providing an open-standards programming model for SOA.

In addition, the WebSphere Application Server Feature Pack for SCA V1.0.1 supports WebSphere Application Server's unique focus on investment protection. In any economy businesses need to make decisions that make use of investments from the past to respond to the future. The SCA feature pack helps businesses protect new IT investments and reuse existing assets in SOA applications.

Designed by key technology vendors, including IBM, the SCA programming model provided in the SCA feature pack supports the principles of SOA through application flexibility and agility. Service implementation and access method details are moved out of business application logic and into metadata that can be operated on by middleware. As service details or locations change, the application remains unaffected. SCA helps application developers maximize productivity through focus on solving business problems with application code rather than protocols and locations.

Highlights of the WebSphere Application Server Feature Pack for SCA are as follows:

- Improves SOA-based composite application time to market and flexibility
- Builds composite applications and services using the open SCA 1.0 programming model
- Uses the reliability and performance of WebSphere Application Server
- Works to protect application investments with a pluggable framework for the latest SCA supported implementation types and protocols

WebSphere Application Server Feature Pack for SCA V1.0.1 capabilities are as follows:

- Create SCA service compositions in POJOs, EJB 2.1-3 components, Java Servlets, and Ajax / JavaScript
- Wire a variety of service types - bindings include web services, JMS, SCA and EJB 2-3
- Expose business logic to web 2.0 style applications through JSON-RPC and ATOM web feeds
- Expose EJB 3.0 and Spring components for re-use in homo or heterogeneous compositions
- Simplified flexible service deployment as a JAR (Java Archive)
- Support for data as Java Architecture for XML Binding (JAXB) or Service Data Objects (SDO) 2.1
- Additional SCA sample applications

IBM WebSphere Application Server Feature Pack for web 2.0

The IBM WebSphere Application Server Feature Pack for web 2.0 extends the reach of SOA.

It provides a supported, open Asynchronous JavaScript and XML (Ajax) development framework that uses existing SOA and JEE assets to deliver rich user experiences.

With Ajax, the interaction model for web applications (such as desktop applications) has become more robust, with continuous interaction and improved usability. Benefits of adding Ajax capabilities to your applications are as follows:

- A more interactive, differentiated experience that can lead to longer sessions and increased customer loyalty.
- Responsiveness, local actions that can result in fewer abandoned transactions, higher completion rates, and higher user productivity.

Unlike other rich web user interface approaches, Ajax applications make use of standard browser features and do not require browser plug-ins.

The WebSphere Application Server Feature Pack for web 2.0 is an IBM-supported solution for creating Ajax-based applications and mashups on WebSphere Application Server. In addition to providing Ajax development tools, this feature pack includes server enhancements to support common web 2.0 applications patterns.

WebSphere Application Server feature packs are optionally installable product extensions that offer targeted, incremental new features and capabilities. For existing WebSphere Application Server customers, the Feature Pack for web 2.0 is available at no additional cost through download for production use. The feature pack for web 2.0 V1.0.1 adds support for the Java EE 6 standard known as JAX-RS or JSR 311. Defined by the Java Community Process, JAX-RS provides a standards based programming model for application developers to build RESTful web services using their existing Java platform investment.

RESTful web services

Driven by developer needs for simplicity and web 2.0 style application requirements, RESTful web services are a popular alternative for exposing business logic in web applications. REST is an architectural style that uses multiple standard technologies such as HTTP, XML, ATOM, and HTML. REST is used to define flexible applications based on the notion of *resources*. A resource is any data that you want to share on the web that you can identify by a Uniform Resource Identifier (URI).

Apache Wink

Apache Wink is an open source software project hosted by the Apache Software Foundation that provides an implementation of JAX-RS. In addition, Apache Wink provides key capabilities to enable common application development needs beyond that which is defined in the JAX-RS standard.

Highlights of the IBM distribution of Apache Wink are as follows:

- JAX-RS 1.0 server run time
- Stand-alone client API with the option to use Apache HttpClient 4.0 as the underlying client
- Built-in support for JavaScript Object Notation library (JSON4J)
- An Atom JAXB model in addition to Apache Abdera support
- Multipart content support
- A handler system to integrate user handlers into the processing of requests

Web 2.0 to SOA connectivity

Web 2.0 applications extend the value of SOA through rich user experiences and integration between external and internal content. As a result of technical and browser security limitations, basic connectivity between Ajax clients and SOA services has been complex and expensive to develop. Web 2.0 to SOA Connectivity reduces costs and speeds up development by simplifying extension of SOA with Ajax.

The WebSphere Application Server Feature Pack for web 2.0 includes an Ajax proxy component that eliminates browser security concerns with cross-domain scripting when combining internal and external services. Also, JSON (JavaScript Object Notation) libraries and web remoting capability simplify connecting directly to JEE services using REST. To extend SOA data outside of the enterprise for partners and customers, this feature pack provides feed syndication support compliant with standards such as ATOM.

Ajax messaging

Traditionally, web applications have relied on a request/response model where user action dictated updates to the browser window. To enable a more interactive user experience, Ajax applications require a different approach.

Ajax Messaging enables a publish and subscribe model where the server can stream data updates, messages, and events in real time to the client. Ajax Messaging uses a Comet application pattern without the need for additional client side plug-ins. The package includes server- and client-side Ajax components that communicate through Ajax friendly JSON-based messages.

The result is that data updates and events can be delivered dynamically to Internet browsers from an enterprise service bus (ESB) or other message source.

Ajax Development Toolkit

With hundreds of proprietary and open Ajax implementations available, developers and architects are faced with yet another tough technology adoption challenge. The question is which Ajax platform emerges as an enterprise standard. IBM, which has adopted the open-source Dojo toolkit (dojotoolkit.org) as its internal standard, is a key contributor to the Dojo project, and a committed member of the Dojo Foundation. The IBM commitments to the Dojo project include enterprise features such as internationalization, data binding, and accessibility support.

For WebSphere Application Server V6.0, V6.1, V7.0, and WebSphere Application Server Community Edition V2.0 and V2.1 customers the WebSphere Application Server Feature Pack for web 2.0 provides the Dojo Toolkit Ajax libraries and web 2.0 sample applications.

By providing a supported Ajax toolkit for WebSphere customers, the IBM Ajax Development Toolkit can reduce time to market and help lower Ajax adoption costs. Additionally, the Feature Pack for web 2.0 works with existing developer IDEs such as IBM Rational Application Developer 7.5.

Note: Two previously available feature packs for WebSphere Application Server V6.1 have been integrated into WebSphere Application Server V7.0. Therefore you can use their features without additional installation procedures. The integrated features packs are as follows:

- WebSphere Application Server Feature Pack for Web Services
- WebSphere Application Server Feature Pack for EJB 3.0

3.3 Development and deployment tools

The WebSphere Application Server V7.0 environment comes with a rich set of development tools. All editions of WebSphere Application Server V7.0 include a full licensed version of the Rational Application Developer Assembly and Deploy V7.5, and a 60-day trial version of the Rational Application Developer for WebSphere Software V7.5.

Rational Application Developer Assembly and Deploy V7.5 supports only the version of the WebSphere Application Server with which it ships. This means that Rational Application Developer Assembly and Deploy V7.5 supports most new

features of WebSphere Application Server V7.0 and supports it as an integrated test environment. It does not, however, support any of the previous versions of WebSphere Application Server as integrated test environments.

Rational Application Developer for WebSphere Software V7.5 supports all new features of WebSphere Application Server V7.0 and is a fully featured integrated development environment for developing SIP, Portlet, web services, and Java EE applications. It supports previous versions of WebSphere Application Server (V6.0 and V6.1) as an integrated test environment.

Figure 3-9 shows the Rational Application Developer packaging.

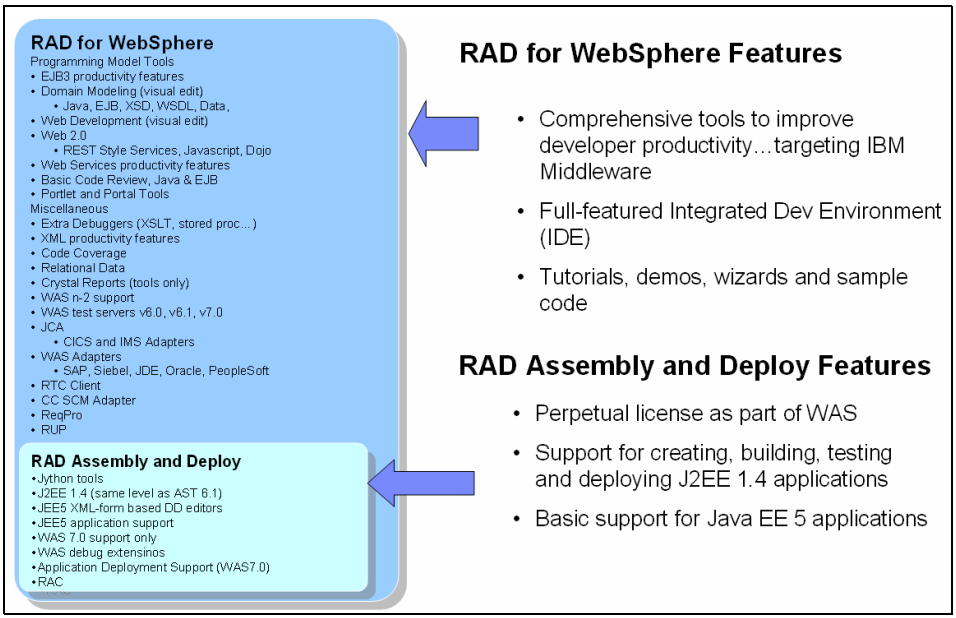


Figure 3-9 Rational Application Developer packaging

3.3.1 Rational Application Developer for Assembly and Deploy V7.5

Rational Application Developer Assembly and Deploy V7.5 is based on the Eclipse 3.4 platform and provides the following features:

- Java EE support
- Development of standard Java 2 Enterprise Edition (J2EE) and Java EE artifacts, such as servlets, JSPs, and EJBs complying with J2EE 1.4 and JAVA EE 5 specifications
- Development of static web projects (HTML, CSS style sheets, JavaScript)

- SIP development, including support for JSR 116 SIP servlets
- XML tools to build and validate XML artifacts, including schemas, DTDs, and XML files
- WebSphere Enhanced EAR support
- Profile management tool
- Support for WebSphere Application Server V7.0 test environments in either a local or remote configuration, but no support for any previous versions of WebSphere Application Server (such as V6.0 or V6.1)
- Jython script development, including script debugging capabilities
- Jacl to Jython script conversion tools (jacl2jython)
- Integration with Concurrent Versions System (CVS), which is a popular Source Code Management (SCM) repository and Rational ClearCase®

Rational Application Developer Assembly and Deploy V7.5 is a development environment that provides you with the tooling necessary to create, test, and deploy the various artifacts supported by WebSphere Application Server V7.0.

It does not include the productivity-enhancing features and visual editors found in Rational Application Developer for WebSphere Software V7.5. It also does not include Rational ClearCase, Crystal Reports, UML modeling, Struts, or JSF support, and it does not support any of the previous releases of WebSphere Application Server (such as V6.0 or V6.1) as test environments. See Table 3-5 for the support level information.

Table 3-5 Specification level support

Specification	Support levels
EAR	1.2, 1.3, 1.4, 5.0
EJB	1.1, 2.0, 2.1, 3.0
WAR	2.2, 2.3, 2.4, 2.5
Application client	1.2, 1.3, 1.4, 5.0

3.3.2 Rational Application Developer for WebSphere Software V7.5

Rational Application Developer for WebSphere Software V7.5 includes all the features of Rational Application Developer Assembly and Deploy V7.5 and adds more productivity-enhancing features, which makes it an even more appealing development environment for advanced Java EE 5 development.

In addition to the features in Rational Application Developer Assembly and Deploy V7.5, Rational Application Developer for WebSphere Software V7.5 brings the following features:

- Full support for Java EE 5, including EJB 3.0 support
- Portal application development with support for WebSphere Portal V6.0 and V6.1 test environments
- EJB test client for easy testing of EJBs
- Support for annotation-based development
- Web 2.0 development features for visual development of Rich Internet Applications with Asynchronous JavaScript and XML (AJAX) and Dojo
- UML modeling functionality
- Integration with the Rational Unified Process and Rational tool set, which provides the end-to-end application development life cycle
- Application analysis tools that check code for coding practices (examples are provided for best practices and issue resolution)
- Enhanced run time analysis tools, such as memory leak detection, thread lock detection, user-defined probes, and code coverage
- Crystal Reports for developing visual data reports
- Component test automation tools to automate creating tests and building and managing test cases

Rational Software Delivery Platform (SDP): Rational Application Developer for WebSphere V7.5 is a product of the Rational Software Delivery Platform (SDP). Rational SDP is a integrated set of products that share a common platform, built on Eclipse 3.4 framework and supports all phases of the development life cycle, as shown in Figure 3-10.

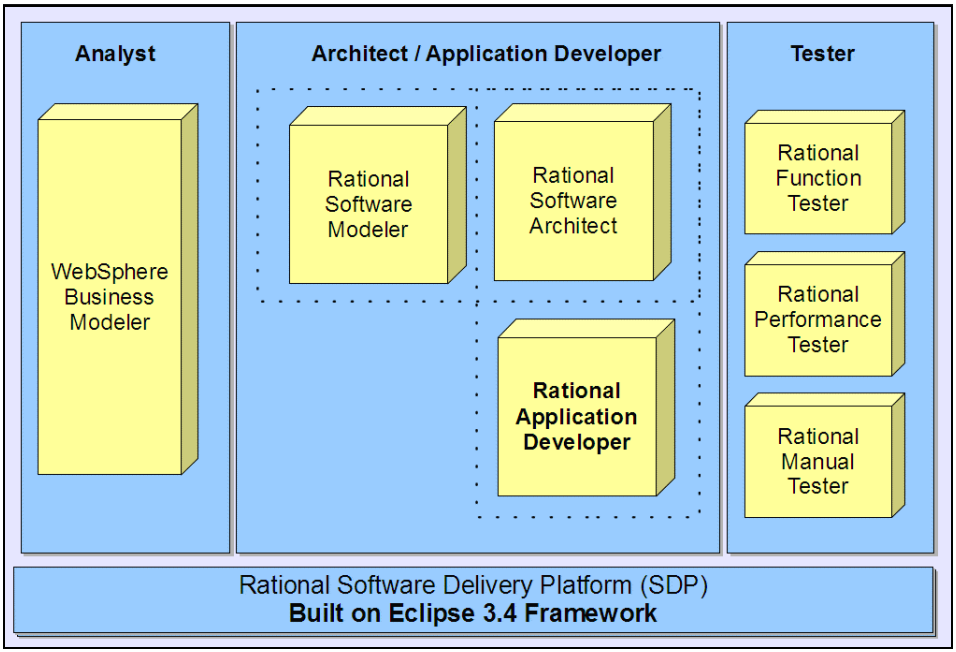


Figure 3-10 Rational Software Delivery Platform overview

3.3.3 WebSphere Rapid Deployment

WebSphere rapid deployment is a set of tools and capabilities (non-graphical command line interface) included in the WebSphere Application Server V7.0 packaging and also used by the development tools. These features allow for the deployment of applications with minimum effort on the part of the developer or administrator. The rapid deployment model has the following three basic features:

- Annotation-based programming
- Deployment automation
- Enhanced EAR file

Annotation-based programming enables you to annotate the EJB, servlet, or web service module code with special Javadoc syntax annotations. When the source of the module is saved, the directives in these annotations are parsed, and the rapid deployment tools use the directives to update deployment descriptors. The developer can then concentrate on the Java source code rather than metadata files.

Deployment automation is where applications installation packages are dropped into a hot directory under an application server and the application is installed automatically. Any installation parameters that have not been specified by the installation package's deployment descriptors have default values that are applied by the automated deployment process.

Rapid deployment allows for a free-form deployment operation. In this mode, it is possible to drop source code or compiled classes for servlets, EJBs, JSPs, images, and so on into the hot directory without strict J2EE packaging. Rapid deployment then compiles the classes, adds deployment descriptors, and generates an EAR file that is automatically deployed on a running server. An enhanced EAR file enables the enterprise archive package to include information about resources and properties, such as data sources, which is required by an application. When deployed on a server, the resources are automatically created on the server.

The WebSphere rapid deployment set of tools can be useful for quickly testing an application. For example, if you know you are going to test several versions of the same application, you can use the automatic deployment feature to have the rapid deployment tools automatically deploy the versions for you. There are limitations and rules you need to follow when working with these utilities, and most of the time you are significantly more productive using a full-blown development environment, such as Rational Application Developer Assembly and Deploy V7.5 or Rational Application Developer for WebSphere Software V7.5

Note: You can use rapid deployment tools for packaging applications at J2EE 1.3 or 1.4 specification levels. The rapid deployment tools do not support Java EE 5 nor J2EE 1.2 specification level.

3.3.4 Choosing which tools to use

Which tool you choose depends on your requirements. If you need to develop, deploy, and test applications on WebSphere Application Server V7.0 for fast turnaround times, choose Rational Application Developer Assembly and Deploy V7.5.

If you are developing applications that require the new features only available in the WebSphere Application Server V7.0, such as JSR 286 events, use Rational Application Developer for WebSphere Software V7.5. It is feature-rich and has lots of productivity-enhancing features not found in the Rational Application Developer Assembly and Deploy V7.5.

4

Migration strategy and planning

This chapter provides relevant information that helps you to build a migration process plan. This is not intended to be a final methodology, but it helps you to organize the migration process, address people about certain tasks, know where you need help or not, and decide how to go through the project.

It provides guidance for you to start the migration and build a project plan and a few artifacts that can be used as input to effect a successful migration.

This chapter is organized in the following sections:

- “Migration process overview” on page 70
- “Before beginning” on page 75
- “Migration planning” on page 77
- “Migration implementation” on page 79
- “Post deployment” on page 83
- “Summary” on page 86

© Copyright IBM Corp. 2010. All rights reserved.

4.1 Migration process overview

There are several factors that influence the decision to perform a migration, including business decisions and technical assessments that leads companies to adapt in a constantly changing and challenging IT environment.

Migration of Java EE applications is not just a change from one application server to another, nor does it mean changing source code. Migration is a process that involves many phases and people from various areas and with various skills. When you migrate to a new application server, you are likely doing it because you want to take advantage of new capabilities, improve the way you manage your applications, refactor your code, improve performance, or decrease downtime.

Although every migration is complex, the best approach is to follow a formal process aimed at evaluating an application development environment, source code, design consideration, and so forth, and makes a recommendation on how a given application is to be migrated to WebSphere environment. You should make these changes an ordered environment, avoiding issues that can be prevented by appropriate of planning and implementing strategies to execute the plan.

Figure 4-1 shows the migration process overview.

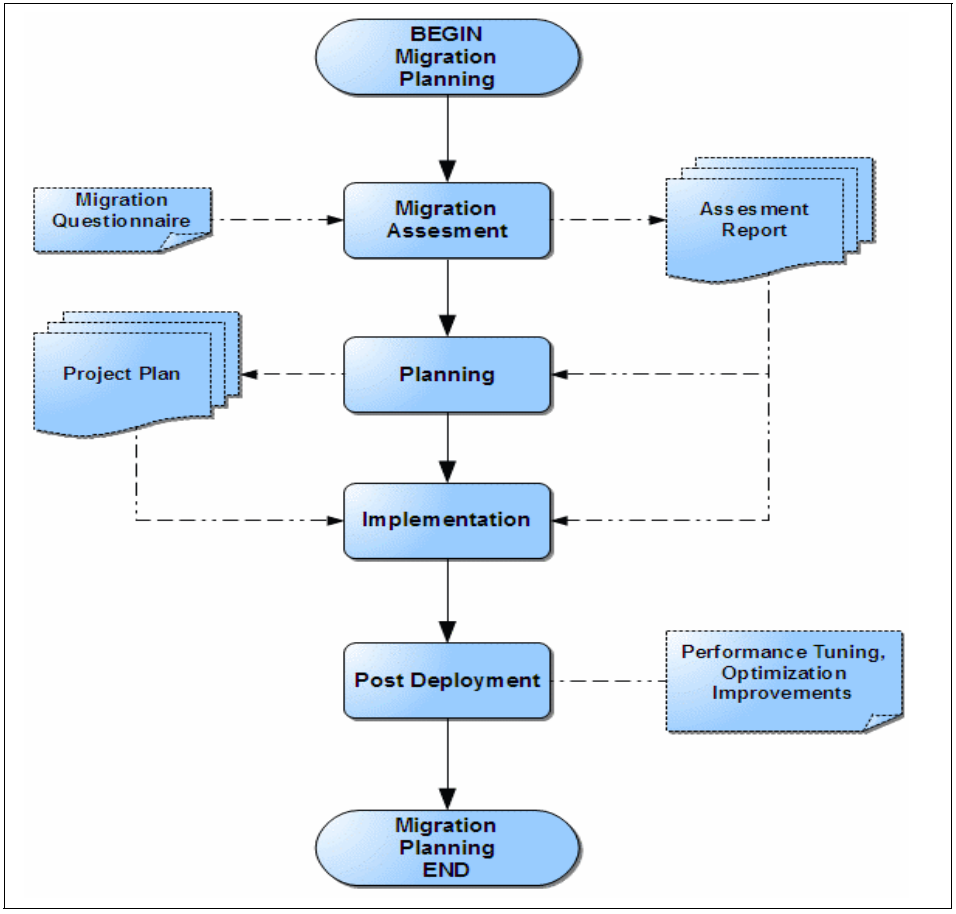


Figure 4-1 Migration process overview

4.1.1 Types of migrations

There are many approaches to take when deciding how the migration will occur. We have identified three that cover pretty much all types of application migrations alternatives. It is a case-by-case decision to on which approach to adopt. Runtime migration is done regardless of type. In each of these, a migration assessment might be done. (migration assessment is discussed later on this chapter)

Depending on the scenario it might be possible that you decide to create a program to manage projects.

Partial

Partial migration consists of migrating multiple or a few applications at a time in a phased approach where the schedule can cover them until the project is completed. It is a good choice when you have teams owning applications in a shared environment or when one team owns multiple applications, which makes it impossible to migrate every application at the same time due to time conflicts.

Applications that are well written and modularized can also be used with this type of migration (for example, moving user interface modules before than business components or vice-versa).

Sample

Sample migration is defined by a migration of a single application that is representative of the other applications. A representative application is the one that has architectural and implementation similarities with most of the applications to be migrated within the enterprise.

This process can be repeated for additional applications after the first one is complete. This is the most common type of application migration because it lays a foundation for future migrations based on challenges encountered in migrating the sample application. You can also determine how quickly migration can be done and highlight all the problems faced so that you can build a concise plan.

Full

Full migration is obvious but has its representativeness. When you decide to go with this option, it is probably because you have a dedicated environment for an application or you are left with no options but to migrate the entire environment due to reasons such as end of support, impossible interoperability of certain applications in old and new environments, and other possible issues.

4.1.2 Important aspects of migrations

Every migrations is different from another. It can vary in terms of complexity, sizing, and other factors. The following list details factors to keep in consideration when planning a migration:

- It is impossible to create a migration roadmap that works for all environments
- Keep the migration as simple as you can, avoiding multiple changes at once
- Do not skip the migration assessment before you begin, regardless of whether it is considered a simple or complex migration
- Do not forget environments other than production, such as development, QA, and testing.

Other considerations are detailed in the next section.

4.1.3 Migration considerations

This section discusses considerations involved with a Java EE applications migration. We are basing comments in the belief that migrations are done to get benefits from newer versions of code, compilers, new feature availability, runtime administration, and performance improvements.

It is important to understand the challenges that you are facing during migration. You can avoid many issues by making informed decisions. To help you in this process, the following list details considerations we discovered in our migrations experiences in IBM and when writing this book:

- Get the applications migrated with smallest number of changes and as quickly as possible.
- Do not skip the migration assessment before you begin, whether or not it is considered a simple or complex migration.
- In the assessment, review the code and evaluate how applications are following or deviating from Java EE specifications.
- Do not forget environments other than production such as development, QA, and testing.
- Even when you know that both application code and application servers (old and new) comply with Java EE specifications, be aware that each vendor can implement in their own way and can also provide extensions to the specification that has to be migrated as well.
- Do not change too much at a time. Use an iterative approach. Migrate a small portion of code and test it, then go back and repeat the process. It is easier to find problems when you have a small controlled environment.
- Keep change variables small. For instance, freeze code changes other than those that are migration-related. Evaluate the changes needed for build scripts and required runtime configurations.
- Determine how to deal with Java SE and Java EE specifications. Should you compile using backward compatibility or upgrade?
- Do not replicate your performance tuning from a previous application server. Use IBM material to make specific performance tuning decisions on WebSphere.
- Get interested parties to discuss the migration and make them interact with each other to learn from previous implementations.
- Never forget that there are tools to help in the migration process. In this book you see them in action.

There are cases where we face external factors that contribute to migration complexity and requires attention from all interested parties. These external factors are integrations with existing systems and interoperability between migrated applications that are dependent of other applications functionality or services.

Interoperability and integrations

Interoperability is the ability of an application, system, or product to interact with another application, system, or product in a way that they can coexist without any special changes. For example, the application you are migrating might interface with an external CRM or ERP system through XML or web services. In this case you need to understand how the application interfaces with those external applications and determine if any change is required for the external application interfaces.

Another example is if you have IBM WebSphere Application Server V7 and Oracle WebLogic Vx in the same network without a firewall between them and you decide to migrate an Oracle WebLogic application to WebSphere Application Server V7. That application has a dependency on an EJB deployed in a second application hosted by the Oracle WebLogic Server. It is probably true that when you migrate the first application to WebSphere Application Server V7, no or minimum changes are required to have them working together. However, be aware that cross-server dependencies during migration can be challenging, as the J2EE specification leaves the implementation of the specifications up to individual vendors.

There are cases where you face problems with this approach (for example choosing sample application migration that is serving components or services to other applications such as EJBs). Later chapters cover common issues like this.

Migration complexity discussions require a good amount of time to decide how to move forward when there are integrations with existing systems. Integration is a sensitive node in the middleware world until now, despite standards, tools, and proven architecture that minimizes its complexity. That is why we are adding this as a tip when managing or participating in a migration project: integrations are not as easy as changing configurations or a few lines of code. Integrations most of the time require revision of standards, compatibility. In many cases the team should consider it as a separate project.

4.2 Before beginning

Sometimes we are tempted to jump into an implementation and planning without enough information to do so, which sooner or later causes undesirable, but predictable, problems. These problems can be better planned with a previous assessment. To avoid that, do not skip the migration assessment.

4.2.1 Migration assessment

This is the starting point of a migration. There is no other way to know the scope, expectations, current environment, teams involved, and all other relevant information to the migration team except by research. The information gathered during the assessment phase helps gain an understanding of the application environment. This information can be used during the planning stage.

This section discusses how the assessment can be conducted and the artifacts that can be produced during this stage.

Migration questionnaires

Because the assessment is one of the first steps in a migration and we jump into a whole new world sometimes with no previous experience from the actual customer. We need to know which kind of information is required to start the project and determine the roles and responsibilities of interested parties. Stakeholders need to be committed to the project to affect a successful migration.

To build this knowledge base we propose that you use what we call migration questionnaires. It is a set of questions to gather necessary information about the migration, business requirements and decisions, hardware and software, topology, architecture, application code, and so forth. During this step, remember to make sure all questions are answered appropriately without generic answers such as “yes” and “no”. For more about migration questionnaire sample questions, see Appendix B, “Migration questionnaires” on page 355.

The IBM WebSphere Application Server Migration Toolkit can be used as a tool when gathering information for the questionnaires. There is a reporting feature that brings out currently known issues that can be exported to PDF report file.

Migration questionnaire review meeting

After collecting required information, typically from separate teams, it is important to discuss it with interested parties. We suggest this review to be made on a regular basis that stakeholders come together to discuss the responses received, resolve issues, and initiate related activities, such as code review and proof of

concepts. These meetings can be done through calls, web conferences, and in-person consultations.

Migration assessment report

At the end of the cycle of questions and answers we need to put together all information received in a report where everyone finds the key indicators that drive decision making. Contents of this report should include sections like: Introduction, Education, Sizing, Topology, Application Migration, Runtime (Server) migration, Testing, Deployment, Tuning and performance, Risks and anything else that you feel are relevant to the project plan.

This report can help you minimize risks and positively affect the success of the project. It answers several questions you might not have considered:

- What is the goal of the project?
- Which benefits does it bring to the business?
- Are developers and administrators familiar with IBM WebSphere Application Server V7?
- Do testers know important areas of testing?
- How many resources and what skills are necessary to execute migration activities?
- How long it can take?
- What is topology that we are planning to use?
- What other environments are involved?
- What are the recommendations for application code migration?
- What configurations need to be done in the runtime migration process?
- Will we create the deployment and administration scripts?

These are just examples of how the report assessment helps the migration, as the better the report, the better the project plan and, probably, the implementation.

4.3 Migration planning

The next step is to plan the migration itself. The assessment report is ready, we know who the stakeholders are, and the size of the work to be accomplished. So what is next?

4.3.1 Roles and responsibilities

In a migration process a key decision is to define who does what. One of the first roles to be assigned is the project manager. The project manager plans and executes the project. Other roles are application owners, developers, testers, administrators, decision makers. Assign appropriate resources to be responsible for certain activities and make them accountable for the tasks assigned to that role.

Usually, the definition of who does the migration is decided in advance, but after the assessment report is created, the plans can change based on what has been found. The assigned team influences the project plan. For instance, depending on who does the migration, more or less time and resources are needed due to education issues or previous experiences, complexity of the project, and so on.

4.3.2 Getting help

There are teams ready to help you in the migration process:

- IBM Business Partners
- IBM Business Partner Services Practise (ISSW)
- IBM Software Services for WebSphere Migration Practise (ISSW)
- Third party sources

IBM migration specialists have been working in these environments for a long time. Expertise in migrations and in the target implementation environment is important and, in a few cases, crucial to minimizing disruption during implementation.

Get support from IBM

Use the following channels to get support and assistance from IBM:

IBM WebSphere Developer Services

IBM WebSphere Developer Services provides developer-to-developer technical assistance to IBM Business Partners when developing applications or building solutions involving IBM WebSphere products. If we require any assistance, we can raise a ticket called a problem management record (PMR). A PMR can be raised from the following web page:

http://www.ibm.com/isv/tech/remoteEmail/entryForm.jsp

Passport Advantage

Passport Advantage® provides support for production issues and product bugs. The following web page helps you to open PMRs (or a service request) with Passport Advantage.

http://www-01.ibm.com/software/support/probsub.html

IBM Software Services for WebSphere (ISSW)

IBM Software Services for WebSphere provides hands-on support and skill-transfer activities to help you successfully deploy your new WebSphere software solution. Visit the following web page to gather more information about ISSW,

http://www3.software.ibm.com/ibmdl/pub/software/dw/wes/pdf/services/DevelopDeployFinal.pdf

WebSphere education

WebSphere education helps us to build and enhance our WebSphere skills. It has more than 250 courses across both the WebSphere Software portfolio and SOA. It provides flexible classroom, online, and private courses. The courses are designed by award winning instructors with first-hand product knowledge.

The role-based training path in WebSphere education assists us by defining a path to acquiring skills for specific WebSphere product offerings. Check out the following link to view a list of WebSphere education offerings.

http://www-01.ibm.com/software/websphere/education/

4.3.3 Migration project plan

A project plan for the migration is based on the assessment report. From there you can decide which activities are dependent upon each other, which activities can run in parallel, and the complexity of each piece migrated. Decisions that were taken, decisions that still need to be taken are all part of a list of activities.

We consider that a project manager assigned to any project knows exactly how to deal with activities, but as our audience in this book is flexible, we are listing the sections that a good migration project plan should contain:

- Dates (rollout, milestones)
- Tasks
- Number of resources assigned
- Test plans
- Dependencies
- Risks
- Production rollout
- Meetings and check points
- Education activities

It is not a final list, so you can add or remove sections that fit your needs.

4.4 Migration implementation

Implementation and plan are closely related. In an iterative project, final phases of an iteration are input for coming iterations. This approach helps to adjust any issues encountered in the project plan, make it more accurate, and set expectations for future rollouts.

There are decisions to be made at the planning phase (for example, which application gets migrated first, which type of migration is used [sample, partial, full]) but also the implementation decisions which are more likely technical can be addressed at this time.

4.4.1 Implementation considerations

Consider migrating run time (server) and application in parallel. This way your environment is ready to deploy after you have finished code changes and adjustments. Use the assessment report to accomplish the following tasks:

- Identify possible changes in the environment
- Engage interested parties and communicate with them

Study the run time and integration questionnaires to see how those answers can help you make the implementation a success. For instance, check for topologies and network answers, identify firewall rules that might prevent your application from running appropriately, check if interoperability of servers is required or if integration with existing systems are being well addressed, make sure test plans cover functional and non-functional requirements, and try specific scenarios and separate sources of review to avoid issues when moving your application to production environments.

4.4.2 Implementation phases

Earlier in this chapter we pointed out that the iterative process is a good option for migration projects. It helps reduce risk by successfully navigating critical paths between phases and providing feedback to future iterations. More on common issues are discussed in Chapter 2, “Common migration issues” on page 7.

We are considering here five phases:

Prepare phase

During this phase check for product and application prerequisites. At this time you should have hardware, software, and people available. Set up development environments, schedule meetings, provide education, and get help if needed.

Migrate phase

Start migrating both application code and runtime configurations. Always remember that the assessment report is your reference, a lot effort has been expended to build that and you can find important information regarding decisions taken and sometimes even code review feedbacks stating challenging areas of the migration.

You already have a decision related to migration type, so document everything that is happening and you benefit from your feedback immediately. If you are a project manager or a delegate, your role is important to address issues that are raised putting everyone together or interfacing with other parties such as software and hardware vendors or third party service providers.

The migration and test phases are closely integrated. It might be useful to test small pieces of functionality until you have everything migrated. This approach helps in two ways:

- Challenges of first migrated functionality serve as input to other migrations and speeds up the process.
- Tests run in parallel with code migration and get prepared for QA and load tests after migration is complete.

Test phase

The test phase is dynamic and has intersections with the migrate and rollout phases. There are three main types of tests that are welcome in a migration process:

- Hands on testing, done together with application migration
- Functional/regression testing to make sure all functionality are working as expected
- Load tests to assess whether or not the application/server is performing equal or better than previous environment

Rollout phase

Rollout phase starts after all previous tests are completed and we have ensured that the application is ready to provide the same or better responses than the previous environment. We know that there are issues that are only raised during production. That is why we have plans and the other previous phases to prevent or nullify chances of rollout back outs.

Deploy the application into the production environment, make the tests again, and if everything is okay, then a decision whether to proceed has to be made. Usually these rollouts require a detailed plan with milestones, owners, and check points to put the team together and address any problems or initiate the backout plan.

Rollouts with interoperability and interfaces involved are much harder to implement because of the potential compatibility problems with the existing applications. Be mindful about these challenges and prepare plans to prevent these issues. A good approach is to have this migration occur during a maintenance window so that you can stop external inputs to the systems affected and test integration and interoperability, so as to minimize risks of having tests mixed with actual valid requests. A clean up of databases or message queues might also be needed.

Cleanup phase

This phase starts when we have successfully deployed the migrated application into production. There are tools that help to monitor performance and system stability that are useful at this moment. Assign specialists to make the appropriate tuning of settings that might be required. Make the changes in the non-production environments before you make them in the production environment and run through the test plans again.

4.4.3 Application migration

There are two distinct migrations. One is related to the server run time from JBoss Application Server to IBM WebSphere Application Server V7, and the other one is related to the application running on top of the application server.

Migration of Java EE applications vary in complexity depending on a series of factors that might have been identified during the migration assessment:

- Source and target Java EE specifications (J2EE 1.4 to Java EE 5)
- Usage of specific Java components (EJB, JSPs, Taglibs)
- Usage of proprietary code - specific for an application server
- Vendor-specific deployment descriptors
- Extended capabilities
- Vendor-specific server configurations (JMS providers, JDBC connection pooling)
- Class loading issues
- Integration with complex external systems such as a CRM or SAP.

These changes can be done manually or by using tools to speed migration. IBM Rational Application Developer (RAD) V7.5 has extended functions to help you in that matter. Additionally, a new migration toolkit has been released that helps you migrate applications from Oracle WebLogic and JBoss application servers to IBM WebSphere Application Server V7. For applications that make use of annotations or want to keep this approach, RAD can provide them through WRD tags to build and maintain applications that are using EJB 3, web services, and other components. Other tools are also available to provide you guidance in code reviews.

Application migration has common steps to be followed. Details on technical activities are covered in other chapters:

- Chapter 2, “Common migration issues” on page 7
- Chapter 6, “Migration from Oracle WebLogic” on page 99
- Chapter 7, “Migrating from JBoss” on page 199
- Chapter 8, “Migrating from Apache Tomcat” on page 275
- Chapter 9, “Migrating from GlassFish” on page 301

4.4.4 Runtime migration

Sometimes we focus and concentrate only in code changes and then when we realize that there are also migration challenges in other areas, it might be too late. In a complex or simple environment there is always something to change, implement, or migrate. Specifically, with migration of Java EE application servers, though compliant with specifications, each vendor has its own way to implement and provide services and improve capabilities.

Considering that the application server is a product that runs under a operating system that can also be part of the migration, be mindful about the changes and configurations to be made at both OS and Application Server levels. Migrations of run time can include the following elements:

- Administrative scripts changes or replacement
- Resources configuration to support application server functionality
- Operating system prerequisites
- Firewall rules
- Security configurations and authentication databases
- Deployment bindings
- Start/Stop scripts
- JVM special or additional configurations
- classloader configurations
- Libraries
- Pooling and thread configurations

It also contains specific tasks from product-to-product. We try to cover those in our product-specific chapters.

4.5 Post deployment

The new environment at least performs as good as the previous one, but we always expect more than that. To achieve this goal we have work to do. This work is done in two fronts that we have already mentioned: application and run time.

Relative to code changes we think that avoidance of unnecessary modifications is more prudent. Get your applications migrated with the smallest number of changes. Unfortunately we do not always have the choice of making changes. Some make the application and the new environment look like they were planned to be: clean, right and fast.

4.5.1 Performance tuning and optimization

Performance tuning and optimization are activities that require expertise and information that support decisions to be taken. This information has two sources: application code review and load test results. With these results in hand, the experts are provided with enough information to make the necessary adjustments. Again, an iterative process helps at this moment. Take small pieces of code or specific configurations and focus on those until you find expected results. This way you have passed through critical paths that lead you to a better performance for the entire application or environment.

This process of having better performance must be conducted carefully, following the project plan. Try to stay at least with what you had before, then after the migration is deployed in production you can see how much you can improve performance. There are tools that help you to get there. IBM provides help through separate channels. Check the WebSphere Application Server V7 Information Center web page to get started:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/welc_howdoi_tprf.html

4.5.2 WebSphere Application Server V7: New opportunities

Using the Java Platform, Enterprise Edition (Java EE) architecture, you can build distributed web and enterprise applications. This architecture helps you focus on presentation and application issues, rather than on systems issues.

Using Rational Application Developer V7.5, you can use the Java EE tools and features to create applications that are structured around modules with differing purposes, such as web sites and Enterprise Java beans (EJB) applications. When you use EJB 3.0 components, you can create a distributed, secure application with transactional support. When you develop applications that access persistent data, you can use the new Java Persistence API (JPA). This standard simplifies the creation and use of persistent entities, and adding new features. For developing presentation logic, you can use technologies such as Java Server Pages (JSP) or Java Server Faces (JSF).

Using the Java EE Platform Enterprise Edition (Java EE), you can develop applications more quickly and conveniently than in previous versions. Java EE 5 significantly enhances ease of use providing

- Reduced development time
- Reduced application complexity
- Improved application performance

Java EE provides a simplified programming model, including the following tools:

- Inline configuration with annotations, making deployment descriptors now optional
- Dependency injection, hiding resource creation, and lookup from application code
- Java persistence API (JPA) allows data management without explicit SQL or JDBC
- Use of plain old Java objects (POJOs) for Enterprise Java beans and web services

Java EE provides simplified packaging rules for enterprise applications:

- Web applications use WAR files
- Resource adapters use RAR files
- Enterprise applications use EAR files
- The lib directory contains shared JAR files
- A JAR file can be specified in the application.xml of the EAR either as an application client or as an EJB module
- A JAR file not specified by the application.xml of the EAR is defined as follows:
  - A JAR file with an application-client.xml implies an application client
  - A JAR file with an ejb-jar.xml implies an EJB module
  - A JAR file with META-INF/MANIFEST.MF specified Main-Class implies an application client
  - A JAR file with any @Stateless, @Stateful, or @MessageDriven annotations implies an EJB application
- A JAR file with Main-Class implies an application client
- A JAR file with @Stateless annotation implies an EJB application
- Many simple applications no longer require deployment descriptors, including
  - EJB applications (.JAR files)
  - Web applications that use JSP technology only
  - Application clients
  - Enterprise applications (.EAR files)

Java EE provides simplified resource access using dependency injection:

- In the Dependency Injection pattern, an external entity automatically supplies an object's dependencies.
  - The object need not request these resources explicitly.
- In Java EE, dependency injection can be applied to all resources that a component needs.
  - Creation and lookup of resources are hidden from application code.
- Dependency injection can be applied throughout Java EE technology:
  - EJB containers
  - Web containers
  - Clients
  - Web services

For additional information about Java EE, see the official specification: JSR 244: Java Platform, Enterprise Edition 5 (Java EE 5) Specification.

http://jcp.org/en/jsr/detail?id=244

Rational Application Developer V7.5

The tool chosen in this book to help most of the migration samples is Rational Application Developer V7.5. We encourage you to look how Rational Application Developer V7.5 can help you improve and migrate from previous Java EE specifications and also take advantage of WebSphere Application Server V7 new features. For more details see the following IBM Redbooks publications:

- *Experience JEE! Using Rational Application Developer V7.5*, SG24-7827
- *Getting Started with WebSphere Application Server Feature Pack for Service Component Architecture*, REDP-4633

4.6 Summary

This process and activities are not supposed to be a final methodology or even the only way to go through a migration project, but rather, it helps you to be prepared and anticipate challenges to come. Following this process as a base for your own projects definitely helps. As you can see, there are several factors that influenced your planning, but basically two things are essential: A migration assessment with the assessment report as input to the project plan and use of a iterative process where first waves or iterations serve as a knowledge base for future ones. Another thing that is important to note is that the approach of migrate first and improve later is a good way to separate pitfalls of both activities although they are somehow related.

5

Installation and configuration

In this chapter, we provide an overview of the environment used in the migration examples, describing the hardware and software used.

We also explain how to install and configure the IBM products used in the migration. This is not intended to be a detailed installation guide. It does, however, provide a step-by-step description of how we installed the products, allowing you to follow the migration examples explained in this book.

This chapter is organized in the following sections:

- "Hardware and software" on page 88
- "IBM Rational Application Developer for WebSphere Software" on page 88
- "IBM DB2 Server Express Edition" on page 97

© Copyright IBM Corp. 2010. All rights reserved.

5.1 Hardware and software

The following hardware and software were used to build our ITSO migration labs:

- Lenovo ThinkCentre with 2.40 GHz Intel Core 2 CPU, 4 GB RAM and 80 GB HDD
- Microsoft Windows Server 2003 Standard Edition with Service Pack 2
- IBM Rational Application Developer for WebSphere Software version 7.5
- IBM WebSphere Application Server 7.0
- IBM DB2 Express 9.7
- IBM WebSphere Application Server Migration Toolkit 1.1

5.2 IBM Rational Application Developer for WebSphere Software

In this section we describe how we installed IBM Rational Application Developer for WebSphere Software.

Download a trial version of IBM Rational Application Developer for WebSphere Software from the following web page:

`http://www.ibm.com/developerworks/downloads/r/rad/`

The following list provides the high-level steps we used to install our IBM Rational Application Developer for WebSphere Software environment.

Installation process: A new installation of IBM Rational Application Developer for WebSphere Software is a two step process. The installation program first installs IBM Installation Manager, and then IBM Rational Application Developer for WebSphere Software.

1. Start the IBM Rational Application Developer for WebSphere Software installation program called `launchpad.exe`.
2. Select the option to install IBM Rational Application Developer for WebSphere Software. A page showing selected packages is displayed, indicating IBM Installation Manager package is installed. Click **Next**.
3. On the Licence Agreement page, select **I accept the terms in licence agreement** after reviewing the License Agreement and Non-IBM Terms. Click **Next**.

4. Choose an installation directory for Installation Manager. We accepted the default installation location:

   ```
   C:\Program Files\IBM\Installation Manager\eclipse
   ```

   Click **Next**.

5. On the Summary page, click **Install**. A confirmation window is displayed, informing you that installation of IBM Installation Manager was completed.

6. Click **Restart Installation Manager** to continue with installation of IBM Rational Application Developer for WebSphere Software. At this point installation of IBM Installation Manager is complete.

7. On the IBM Installation Manager welcome page, select the option to install software packages.

8. On the Package Selection page, select the options for IBM Rational Application Developer for WebSphere Software and IBM WebSphere Application Server Version 7.0 Test Environment. Click **Next**.

9. On the License Agreement page, review the License Agreement and Non-IBM Terms. Select **I accept the terms in licence agreement**. Click **Next**.

10. Choose an installation location for shared resources. We accepted the default installation location:

    ```
    C:\Program Files\IBM\SDPShared
    ```

    Click **Next**.

11. Choose an installation location for IBM Software Delivery Platform package group. We accepted the default installation location:

    ```
    C:\Program Files\IBM\SDP
    ```

    Click **Next**.

12. On the following page, click **Next** to continue without selecting the option to extend an existing Eclipse installation.

13. On the Language Selection page, select your language and click **Next**.

14. On the Feature Selection page, click **Next** to continue with default options.

15. On the Common Configurations page, click **Next** to continue with default options.

16. On the page for configuration of IBM WebSphere Application Server Version 7 Test Environment, select a profile name and location. We accepted the default profile name and profile path.

    ```
    was70profile1
    C:\Program Files\IBM\SDP\runtimes\base_v7\profiles\was70profile1
    ```

    Fill in your user ID and password for administrative security options. Click **Next**.

17. On the summary page, review the information presented and click **Install**.
18. The confirmation page informs you that installation is completed. Click **Finish**.
19. The installation program launches IBM Rational Application Developer for WebSphere Software. A dialog box is displayed, asking you to choose a workspace location. We accepted the default location. Click **OK**.
20. After IBM Rational Application Developer for WebSphere Software is launched, the Administrative Settings dialog box is displayed. Enter the same user ID and password you used at step 16 on page 89 and click **Finish**.
21. License Management dialog box is displayed, informing you on details of your current licence. Click **Done**.

IBM Rational Application Developer for WebSphere Software Version 7.5 is now installed, with an integrated IBM WebSphere Application Server Version 7.0 Test Environment.

5.2.1 Installing the latest fixpack

Package updates provide fixes and updates to installed product packages. You can use the Update Packages wizard in IBM Installation Manager to install updates for product packages that were installed by using IBM Installation Manager.

Online update

If you have access, this is the simplest method for a single installation of IBM Rational Application Developer for WebSphere Software. Using online updating, update IBM Installation Manager, IBM Rational Application Developer for WebSphere Software, and the integrated test environments.

For detailed instructions on using this method, see the topic “Updating Installed Product Packages” in the information center for IBM Rational Application Developer for WebSphere Software, at the following web page:

`http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/index.jsp?topic=/com.ibm.rad.install.doc/topics/t_update.html`

High-level steps are as follows:

1. Close any running instances of IBM Rational Application Developer for WebSphere Software.
2. Start IBM Installation Manager by navigating to **Start** → **Program Files** → **IBM Installation Manage** → **IBM Installation Manager**.
3. On the Start page of IBM Installation Manager, click **Update**.

Important: At the time of writing, the latest update of IBM Rational Application Developer for WebSphere Software requires an updated version of IBM Installation Manager. Continue with the update of Installation Manager. Follow the instructions in the wizard to complete the update.

4. In the Update Packages wizard, select **IBM Software Delivery Platform** and click **Next**. Installation Manager searches for updates in its repositories and the predefined update sites for the product.
5. On the Update page, keep the default selections. Click **Next**.
6. On the Licenses page, after reviewing the license agreements for the selected updates, click **Next**.
7. On the Features page, keep the default selections and click **Next**.
8. On the Summary page, click **Update** to download and install the updates. A progress indicator shows the percentage of the installation completed.
9. Click **Finish** to close the wizard.

Offline update

The steps to perform an offline update are basically same as for an online update, but you must download the updates and register them as repositories with IBM Installation Manager first.

All updates can be downloaded from the following web page:

`http://www.ibm.com/software/support`

Important: If any of the updates require a more recent version of IBM Installation Manager than the one currently installed, updates cannot be applied before IBM Installation Manager itself is updated first.

The following updates must be downloaded separately and the following process must be repeated for each of them:

1. Update IBM Installation Manager.
2. Update IBM Rational Application Developer for WebSphere Software.
3. Update IBM WebSphere Application Server Integrated Test Environment.

To install updates from the compressed files, complete the following steps:

1. Extract the compressed files in an temporary directory. For example, we chose `C:\RAD_fixes`, and made a separate subdirectory for each upgrade:
   - Installation manager: `C:\RAD_fixes\installation manager fix`
   - IBM Rational Application Developer for WebSphere Software: `C:\RAD_fixes\RAD`
   - Integrated test environment: `C:\RAD_fixes\WAS 70`
2. Add the update's repository location in IBM Installation Manager by performing the following steps:
   a. Start IBM Installation Manager.
   b. On the Start page of Installation Manager, click **File** → **Preferences**, and click **Repositories**. The Repositories page opens.
   c. On the Repositories page, click **Add Repository**.
   d. In the “Add repository” window (Figure 5-1), browse to or enter the file path to the `diskTag.inf` file (or `repository.config` file when updating IBM Installation Manager itself), which is located in the disk1 sub-directory in the directory where you extracted the compressed files. Click **OK**.

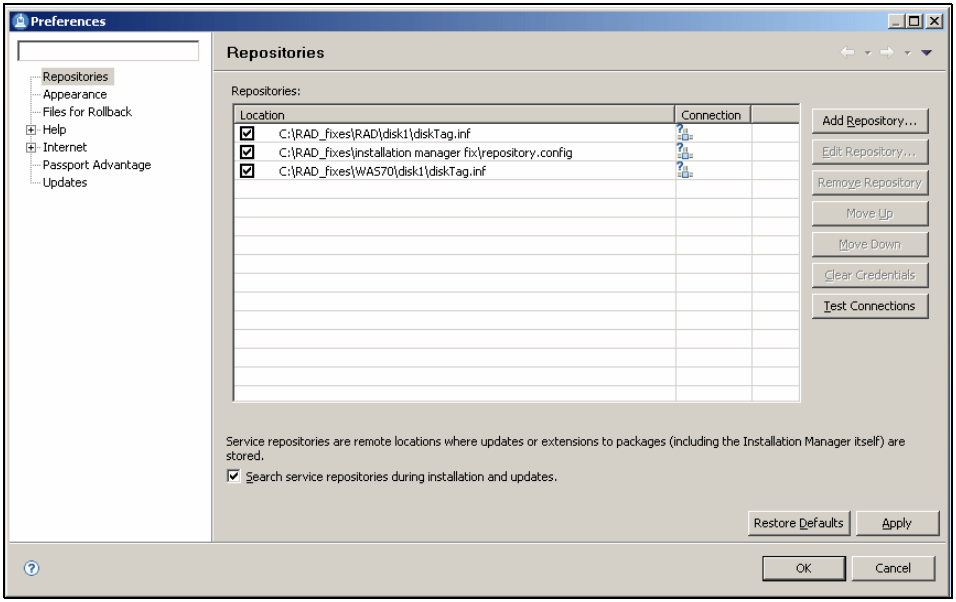


Figure 5-1 IBM Installation Manager preferences dialog after adding repositories

   e. Click **OK** to close the Preference page.
3. Install the update as described in online update section.

5.2.2 Verifying installation

The following steps describe how to verify the updates.

1. Launch IBM Rational Application Developer for WebSphere Software by navigating to **Start** → **Programs** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 7.5** → **IBM Rational Application Developer**.
2. View version information by navigating to **Help** → **About IBM Rational Application Developer for WebSphere Software**. Verify that the correct version is presented.

   At the time of writing, the latest version of IBM Rational Application Developer for WebSphere Software is 7.5.5.1, as can be seen in Figure 5-2.

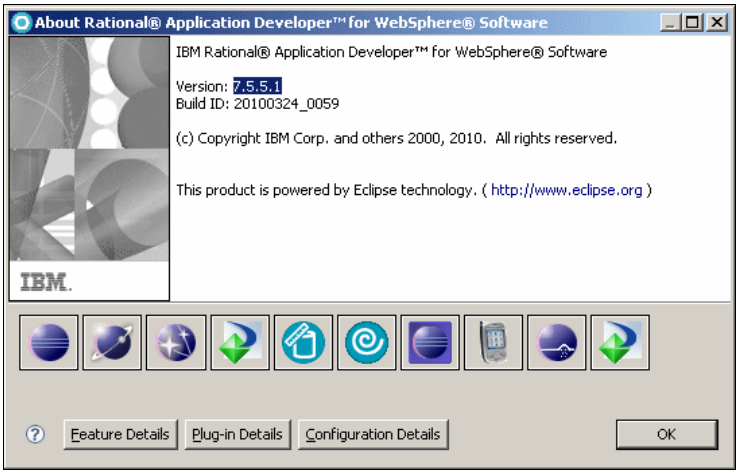


Figure 5-2 RAD version information as displayed in About window.

3. Press the green arrow icon in the Servers tab in the bottom portion of the main IBM Rational Application Developer for WebSphere Software window to launch integrated testing environment. See Figure 5-3.

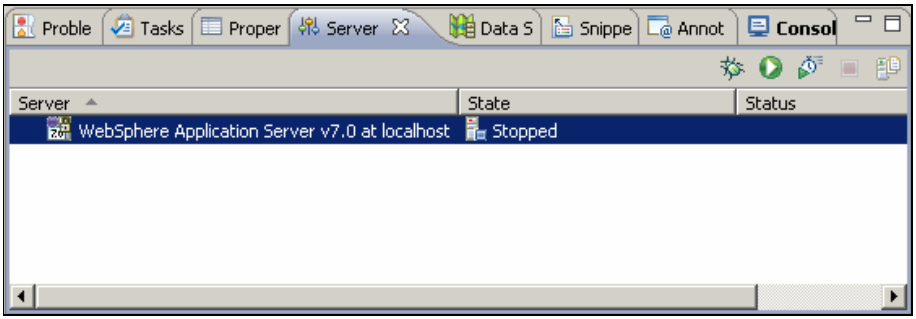


Figure 5-3 Integrated testing environment server view

4. After the status in this window changes to Started, right-click the server name and select **Administration** → **Run administrative console**.
5. An administrative console login window is displayed. Log in with the credentials selected during installation and click **Log in**.
6. On the Welcome page, verify the version of the integrated test environment. See Figure 5-4.

Figure 5-4 Welcome page of test environment showing current version

5.3 IBM Application Migration Tool

In this section, we describe how we installed and configured IBM Application Migration Tool, built on Rational Software Analyzer technology.

Find information and downloads for Application Migration Tool at the following web page:

http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html

The Application Migration Tool is a plug-in that you can install into an existing Eclipse IDE, version 3.4.2 or higher. Download Eclipse IDE from the following web page:

http://www.eclipse.org

Use IBM Rational Application Developer for WebSphere Software and Rational Application Developer for Assembly and Deploy, as these are Eclipse-based products. This is the preferred method, even if the tool can work with a basic Eclipse IDE, as it provides access to other functionality provided by these products.

5.3.1 Installing Application Migration Tool into IBM Rational Application Developer for WebSphere Software

The following steps describe how to install the Application Migration Tool into IBM Rational Application Developer for WebSphere Software:

1. Download the Application Migration Tool from the following web page:

   http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html

   Save the file to a suitable location. We used the `C:\Toolkit_setup folder`.
2. Start IBM Rational Application Developer for WebSphere Software.
3. Open Software Updates and Add-ons window by navigating to **Help** → **Software updates**. Select the **Available Software** tab.
4. Click the **Add Site** button.

5. On the Add Site dialog box, click the **Archive** button. A file dialog box is displayed. Select the downloaded plug-in file and click **OK**. See Figure 5-5.

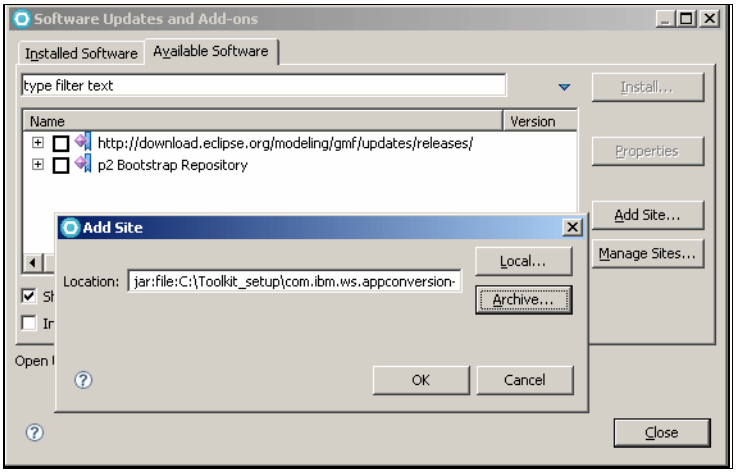


Figure 5-5 Plug-in file

6. The new update site is displayed in the list. Select the **Update Site for Application Migration Tool** and click **Install**. See Figure 5-6.

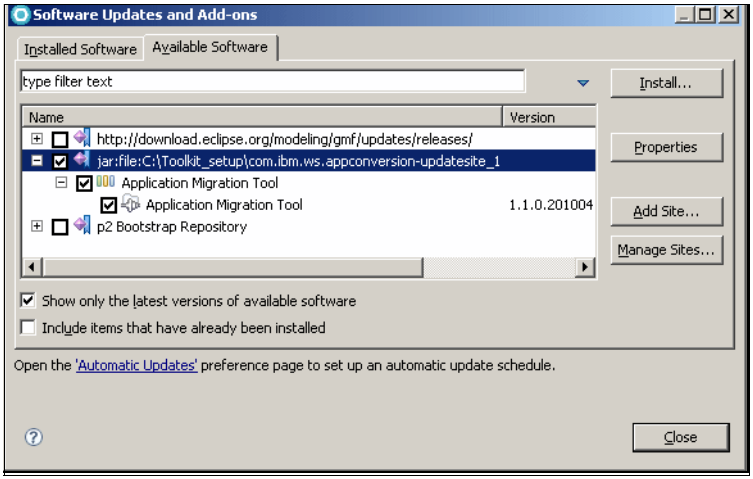


Figure 5-6 Selecting toolkit update site

7. On the install window, click **Next** to continue.
8. On the Review Licences window, select the **I accept the terms of license agreement** option and click **Finish**.

9. When the Software Updates window is displayed, informing you that installation was completed, click **Yes** to restart IBM Rational Application Developer for WebSphere Software.

10. After the restart is completed, to verify the installation, navigate to **Help** → **About IBM Rational Application Developer for WebSphere Software**. Click **Features Details** and scroll down to locate the Application Migration Tool feature, as shown in Figure 5-7.

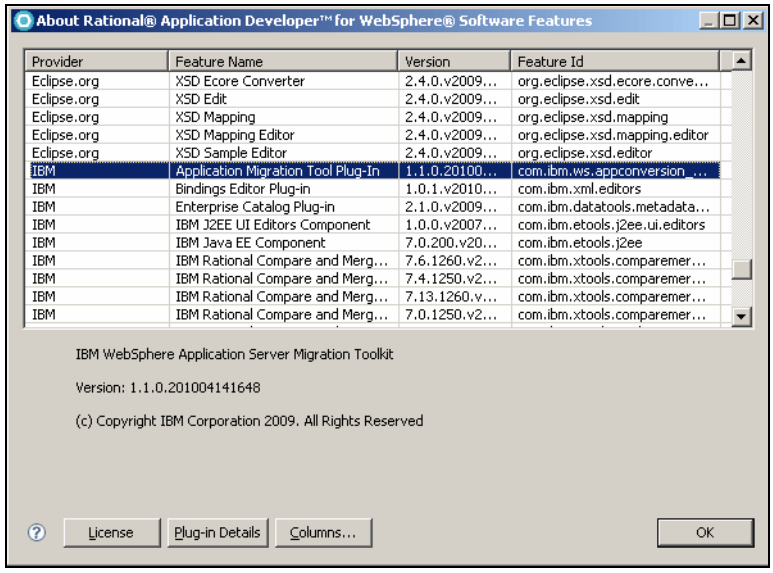


Figure 5-7 Verifying installation of toolkit

5.4 IBM DB2 Server Express Edition

In this section we describe how we installed and configured Version 9.7. For further DB2 installation and configuration instructions, refer to the DB2 Information Center at the following web page:

`http://publib.boulder.ibm.com/infocenter/db2help/index.jsp`

You can download DB2 Universal Database Express Edition at the following web page:

`http://www.ibm.com/developerworks/downloads/im/udbexp`

To install DB2 Universal Database Express Edition, follow these steps:

1. Launch the `setup.exe` setup program.
2. On the DB2 Setup Launchpad, select **Install a product**.
3. Scroll down to locate DB2 Express Version 9.7 and click **Install New**.
4. On the Welcome page, click **Next**.
5. On the Software License Agreement window, select the **I accept the terms in the license agreement** option after reviewing the license and click **Next**.
6. On the Installation Type page, select **Typical installation** and click **Next**.
7. Leave the default response file options on the following page and click **Next** on the following page.
8. On the Installation folder page, select an installation location. We used the default installation location, `C:\Program Files'SQLLIB\`. Click **Next**.
9. On the next page, enter user information for the DB2 Administration Server. Accept the default values and click **Next**.
10. On the Configure DB2 Instances page, accept the default values and click **Next**.
11. Click **Finish** on summary page to start installation.
12. After setup is completed, an information page is displayed. Click **Next**.
13. Click **Finish** to complete installation on the next page without selecting any other additional products to install.
14. The setup program automatically launches the First Steps page upon successful termination. Close this page.

6

Migration from Oracle WebLogic

In this chapter, we describe the migration process and individual migration issues and solutions we found when migrating two J2EE 1.3 applications from Oracle WebLogic Server 9.2 to WebSphere Application Server V7. We provide step-by-step instructions for migrating these applications.

We also discuss the capabilities of the IBM WebSphere Application Server Migration Toolkit V1.1 and the configuration options available to take advantage of the tool's analysis capabilities.

The compatibility of application servers these days is surprisingly good. We did not have major problems with migrating our sample applications. In both cases, we were able to deploy WebLogic compiled EAR files into WebSphere with minimal changes to the Java source code. Most of the changes were limited to deployment descriptors. This is as expected because the J2EE specification allows for vendor-specific deployment descriptors.

This chapter is organized in the following sections:

- "Introduction" on page 100
- "Prerequisites and assumptions" on page 101
- "Oracle WebLogic Server 9.2 installation" on page 102
- "Application Migration Tool overview" on page 103
- "Migrating Trade 3.1 for Oracle WebLogic Server 9.2" on page 104
- "xPetstore EJB migration" on page 166

© Copyright IBM Corp. 2010. All rights reserved.

6.1 Introduction

Oracle WebLogic Server 9.2 was released in October of 2006 and is the J2EE runtime version from Oracle used throughout our migration examples in this IBM Redbooks publication.

During our migration exercise for this platform, we used two J2EE 1.3 sample applications:

- Trade 3.1
- xPetstore EJB 3.1.3

The Trade 3.1 application that we have selected for migration was developed by IBM for performance testing purposes and does not use any non-J2EE features of Oracle WebLogic Server 9.2. We selected Trade for the following reasons:

- It is a well built application
- It uses most of the J2EE 1.3 APIs discussed in 1.3.1, “Java SE and EE technologies covered” on page 4.
- It implements a number of good programming techniques for performance.
- It implements a number of patterns to reuse in your application development efforts.

Note: There is a possibility that your application might have a dependency on WebLogic proprietary extensions such as WebLogic startup and shutdown classes, or it might have been constructed using the WebLogic Workshop development tool. If this is the case, your application is likely not portable to other application servers and requires a redesign and rewrite.

The second application we have migrated is xPetstore EJB 3.1.3, which is a rewrite of Sun’s (now Oracle) Petstore application. This might not be the typical application that you find in the enterprise because it was designed and built to be portable across multiple application server platforms. The reason we have selected this application is that it was not designed for WebSphere and there is no evidence that it has ever been tried in WebSphere with DB2. The application uses a number of elements of J2EE 1.3 discussed in 1.3.1, “Java SE and EE technologies covered” on page 4.

We used the new WebSphere Application Server Migration Tool V1.1 to migrate both the Trade and xpetstore applications. We explain the capabilities of the new migration toolkit and how to configure it in 6.4, “Application Migration Tool overview” on page 103.

Another useful tool that you can use Oracle's Java Application Verification Kit (AVK). This tool is intended to help developers test their applications for the correct use of J2EE APIs and portability across J2EE. See more details at the following web page:

http://java.sun.com/j2ee/verified/avk_enterprise.html

6.2 Prerequisites and assumptions

There are prerequisites and assumptions we need to consider before continuing with the migration examples. One basic requirement is that the reader should have the destination environment installed and configured as described in Chapter 5, "Installation and configuration" on page 87. That chapter provides initial background knowledge if the reader is not yet familiar with IBM products.

The reader should be familiar with the J2EE specification and architecture, and be able to understand and run basic SQL commands. The reader should be familiar with Rational Application Developer V7.5 and Ant. The reader should also have basic familiarity with Oracle WebLogic Server 9.2.

Install the following software before starting the migration:

- Java SE 6

  We use Java SE 6 because it is supported by WebSphere Application Server V7 and we need to make sure our applications work well in this JVM. Prior to Java SE 6, the specification name was Java 2 Platform, Standard Edition (J2SE). The WebSphere Application Server V7 installation includes Java SE 6. You do not need to install additional JDKs on your machine.

- Ant 1.6.5

  Apache Ant is a Java-based build tool. It is similar to the Make tool, but has many enhanced features so it can be extended using Java classes instead of writing shell commands, which makes it a good choice for Java applications.

- IBM WebSphere Application Server V7

  Before migrating to WebSphere Application Server V7, we recommend you explore sample applications shipped with the product and read the publications listed in "Related publications" on page 377.

- IBM DB2 UDB 9.5

  WebSphere Application Server V7 supports many databases, but we decided to use DB2 for all our applications. Instructions on how to install and configure DB2 Universal Database V9.5 can be found in 5.4, "IBM DB2 Server Express Edition" on page 97.

- IBM Rational Application Developer V7.5

  We used the new WebSphere Application Migration Tool V1.1 plugin for Rational Application Developer V7.5 that supports the migration of J2EE applications developed for competitive application servers. Details on the tool capabilities can be found in 6.4 "Application Migration Toolkit Overview".

  Instructions on how to install and configure Rational Application Developer V7.5 can be found in 5.2, “IBM Rational Application Developer for WebSphere Software” on page 88.

6.3 Oracle WebLogic Server 9.2 installation

This section provides high-level steps and tasks for installing and configuring Oracle WebLogic Server 9.2 MP2 (Maintenances Pack 2) on the Microsoft Windows XP Professional platform. Detailed instructions for installing, configuring, and managing Oracle WebLogic Server 9.2 are provided in the product download page available at the following web page:

`http://www.oracle.com/technology/software/products/ias/htdocs/wls_main.html`

The following list highlights the high-level tasks performed to install and configure the initial WebLogic domain environment. This was done to have a clean setup as our starting point for deploying the two sample applications.

1. Download Oracle WebLogic Server 9.2 MP2, which is a suite that includes WebLogic Server 9.2 MP2 and other Oracle products from the aforementioned Oracle web page.
2. Install the software in the `C:\bea` directory. This directory is later referred to as <weblogic_home>.
3. Launch the Oracle WebLogic Configuration Wizard by clicking **Start** → **Programs** → **BEA Products** → **Tools** → **Configuration Wizard**.
4. On the Welcome window, select **Create a new WebLogic domain**.
5. On the Select Domain Source window, select **Generate a domain configured automatically to support the following BEA products**.
6. On the Configure Administrator Username and Password window, enter the username and password of your choice. We used `weblogic/weblogic` in our installation.

7. On the Configure Server Start Mode and JDK window, make the following selections:
   - WebLogic domain startup mode: **Development Mode**
   - JDK: **Sun SDK 1.5.0_10**
8. On the Customize Environment and Services Settings window, keep the default selections and click **Next.**
9. On the Create WebLogic Domain window, enter the following information:
   - Domain name: **TradeDomain**
   - Domain location `D:\bea\user_projects\domains\TradeDomain`
10. Click the **Create** button.
11. Run the configuration wizard a second time and create a second domain for the xPetstore EJB application. This time, create **xpetstore** in the following directory:

    `D:\bea\user_projects\domains\xpetstore`

6.4 Application Migration Tool overview

The IBM WebSphere Application Server Migration Toolkit V1.1 is part of the IBM WebSphere Application Server Migration Toolkit. It is based on IBM Rational Software Analyzer and it provides a way for software developers to migrate their Java J2EE/EE applications from one third-party application server to WebSphere platform. The process of converting a Java applications can involve modifying Java source code, JSP pages, vendor-specific classes, and deployment descriptors, which can be costly and time-consuming process.

The tool flags a number of known differences between applications hosted on Oracle WebLogic or JBoss and WebSphere Application Server. In many cases, the tool can automatically convert these key differences. If the tool is unable to perform the fix, it flags the file in question to provide insight as to where design changes are needed and how to perform a manual fix.

The migration tool supports the following tasks:

- Migrating applications to WebSphere Application Server Version 7
- Migrating WebLogic Java, JSP, and class path artifacts (Java EE 5 and prior versions)
- Migrating WebLogic deployment descriptors (J2EE 1.4 and prior versions)
- Migration JBoss Java and class path artifacts (J2EE 1.4 and prior versions)
- Migrating JBoss deployment descriptors (J2EE 1.4 and prior versions)
- Supports DB2, Oracle, SQLServer and SQLServer2000 databases

The Application Migration Tool V1.1 and associated documentation can be downloaded from the following web page:

http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html

The Application Migration Tool forum is available to provide input and get questions answered from the following web page:

http://www.ibm.com/developerworks/forums/forum.jspa?forumID=2106

6.5 Migrating Trade 3.1 for Oracle WebLogic Server 9.2

The Trade 3.1 application was developed by IBM to evaluate J2EE performance issues. Although built for performance evaluation, the application provides a wide range of features that we were interested in covering during the migration exercise. The latest version of Trade is V6. It is built as a complete J2EE 1.4 application that uses most components of the specification. With the exception of the J2EE level, both versions of the Trade application are functionally equivalent.

We used Trade 3.1 in this book, but if you are interested in experimenting with Trade V6 version, it is available for download from the following web page:

http://www.ibm.com/software/webservers/appserv/was/performance.html

The Trade 3.1 version we used for Oracle WebLogic Server 9.2 is a slightly modified version. See Appendix C, “Additional material” on page 375 for details on how to download the Trade 3.1 used for Oracle WebLogic Server 9.2.

The modifications made to the application were to configuration files and deployment descriptors. The core of the application remains unchanged, so we do not expect any issues with the Java code. We mainly focus on migrating deployment descriptors and creating resources in WebSphere Application Server V7 to support the application.

The main features covered and provided by this application make extensive use of the following J2EE 1.3 APIs:

- JMS 1.1
- JDBC 2.1
- Entity and Stateless Session EJB 2.0
- Primary Key generation pattern for CMP EJB 2.0
- Message Driven beans 2.0
- 2-phase commit transactions
- JSP 1.2
- Servlet 2.3

Figure 6-1 shows the Trade3 J2EE components.

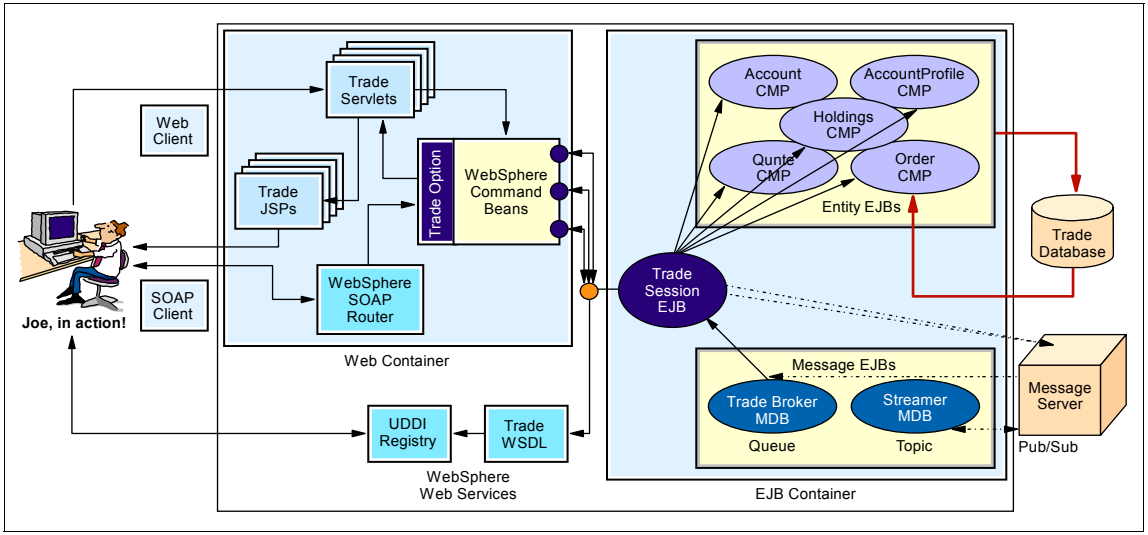


Figure 6-1 Trade3 J2EE components

6.5.1 Migration approach

We decided to perform the Trade migration using Rational Application Developer V7.5.5. In addition to Rational Application Developer V7.5, we use the Application Migration Tool V1.1. We decided to use the migration tool because it supports the migration of EJB deployment descriptors, Java source code, and JSP code from Oracle WebLogic Server 9.2 to WebSphere Application Server V7.

The migration is performed in the following steps:

1. Verify that application works in its original environment. In this case, we deploy and run the Trade 3.1 application in Oracle WebLogic Server 9.2.
2. Import the EAR file with source code into Rational Application Developer V7.5.
3. Configure and run the IBM WebSphere Application Server Migration Toolkit V1.1.
4. Fix errors reported by the IBM WebSphere Application Server Migration Toolkit V1.1.
5. Migrate the EJB deployment descriptors using the IBM WebSphere Application Server Migration Toolkit V1.1.

6. Build Trade for WebSphere Application Server V7 using Rational Application Developer V7.5.
7. Deploy and test the application on WebSphere Application Server V7 and identify and solve any problems reported.

6.5.2 Configuring the initial environment

After installing Oracle WebLogic Server 9.2 and creating a new server configuration, deploy the sample application to the initial environment. This section describes the procedures we followed for creating a sample database, configuring Oracle WebLogic Server 9.2 to hold the Trade application, deploying the application, and testing.

Due to the size of this section, the following list details the subsections it contains:

- "Acquiring the Trade application" on page 106
- "Creating the sample application database" on page 107
- "Adding the DB2 JDBC driver to the WebLogic class path" on page 108
- "Creating a new data source" on page 109
- "Creating JMS resources" on page 110
- "Deploying the sample application and populating the database" on page 112

Acquiring the Trade application

See Appendix C, "Additional material" on page 375 for details on how to download the Trade 3.1 application for Oracle WebLogic Server 9.2.

Once you have downloaded the application, create a Trade3 directory and extract the downloaded file with the name `trade3-WebLogic.zip`. The compressed file contains the following files and directories:

- `db2_schema.ddl`

  Script for creating the database and tables in DB2 Universal Database V9.5.

- `trade3wls.ear`

  Trade application binaries.

- trade3EJB

  Directory containing source code for the EJB module.

- trade3WSAppClient

  Directory containing the source code for a stand-alone client that uses web services to communicate with Trade. The source code is only needed at compile time. The compiled classes are not included in the EAR file and are not a runtime dependency, which we notice later when starting the migration.

- trade3AppClient

  Directory containing source code for the J2EE client application for Trade.

- trade3Web

  Directory containing the source code for the web module.

In our scenario, we used the `D:\sampleApp\Trade3` directory to extract the application. It is also referred to as `<source_home>`.

Creating the sample application database

In this section, we describe the steps for creating the Trade database. The database schema is included in the in `trade3-WebLogic.zip` file.

1. From a command line, start the DB2 CLP (command line processor) as follows:

   ```
   cd <source_home> (i.e. cd D:\sampleApp\Trade3)
   db2cmd
   ```

2. Execute the following commands in DB2 command prompt to create the Trade database and tables:

   Note: The user you use to connect to the database defines the name of the schema where the DDL creates the tables.

   ```
   db2 create db trade3db
   db2 connect to trade3db user db2admin using its0ral0
   db2 -tvf db2_schema.ddl
   db2 disconnect all
   db2 update db config for trade3db using logfilsiz 1000
   db2 update db cfg for trade3db using maxappls 100
   db2stop
   db2start
   db2 connect to trade3db user db2admin using its0ral0
   cd <db2_home>\bnd
   db2 bind @db2cli.lst blocking all grant public
   ```

At this point, the database is empty and ready to get the data loaded. The creation of initial data in the database is done by the application when it is deployed.

Adding the DB2 JDBC driver to the WebLogic class path

Earlier, we created a new WebLogic configuration to have a clean environment to start deploying and configuring the Trade application. The creation of this new configuration is an optional step. You can use any of your existing WebLogic domains for this task.

For the new WebLogic configuration, update the `startWebLogic.cmd` file located at `c:\bea\user_projects\domains\TradeDomain` by adding the location of the DB2 JDBC drivers to the end of the class path. Make sure that you specify both `db2java.zip` and `db2jcc.jar` and db2 license files as shown in Example 6-1.

Example 6-1 Excerpt from the startWebLogic.cmd configuration file

```
set DB2=C:\IBM\SQLLIB\java
set
JDBC_TRADE_CLASSPATH=%DB2%\db2java.zip;%DB2%\db2jcc.jar;%DB2%\db2jcc_li
cense_cisuz.jar;%DB2%\db2jcc_license_cu.jar
set
CLASSPATH=%CLASSPATH%;%MEDREC_WEBLOGIC_CLASSPATH%;%JDBC_TRADE_CLASSPATH
%
```

In our scenario, the DB2 JDBC drivers are located in the `C:\IBM\SQLLIB\java` directory.

3. Start the Oracle WebLogic Server 9.2 instance from the command line:

   ```
   d:
   cd \bea\user_projects\domains\TradeDomain
   startWebLogic.cmd
   ```

4. Access the Console Login when the server is up and running. When we created the new WebLogic configuration, we accepted the default values for port and location. In our scenario the Console Login can be accessed at the following URL:

   `http://localhost:7001/console`

   The username and password required for login were specified during the creation of the new WebLogic configuration. In our scenario, we used `weblogic` for both the username and password.

Creating a new data source

Perform the following steps to create a new data source:

1. On the WebLogic Server Console, navigate to **TradeDomain** → **Services** → **JDBC** → **Data Sources** and create a new data source with the characteristics listed in Table .

Table 6-1 Data source values

Parameter	Value
Database type	DB2
Database Driver	IBM DB2 Driver (Type 2 XA) Versions:7.X, 8.X
Name	TradeDataSource
JNDI Name	jdbc/TradeDataSource

Click **Next**. The Transaction Options window is displayed and an informational message is displayed stating that based on the selected driver type, only XA 2 phase commit transactions supported. Click **Next**.

2. On the Connection Properties window, enter the information listed in Table 6-2.

Table 6-2 Database Connection Properties Values

Parameter	Value
Database Name	trade3db
Host Name	localhost
Port	50000
Database User Name	db2admin
Password	db2adm1n

Note: Use the database user name and password for your DB2 installation. In our example we used `db2admin`/`db2adm1n` respectively.

3. On the Test Database Connection page, click the **Test Configuration** button. You should see the message `Connection test succeeded`.
4. On the Select Targets window, select **AdminServer** as the target server and click **Finish**.

Creating JMS resources

Use the following steps to create the JMS resources:

1. From the WebLogic Server Console home page, create a new JMS Server by navigating to **TradeDomain** → **Services** → **Messaging** → **JMS Servers**. Use the parameters in Table 6-3 to create a new JMS Server.

Table 6-3 JMS Server values

Parameter	Value
Name	TradeJMSServer
Target	AdminServer

2. Create a JMS module to act as the container to defining JMS queues, topics, connection factories.
3. Create a JMS Module by navigating to **TradeDomain** → **Services** → **Messaging** → **JMS Modules**.
4. Enter **JMS Resource** for the JMS Module name and click **Next**.
5. Select **AdminServer** as the target and click **Finish**.
6. Create the JMS queues, topics, and connection factories needed by the Trade application at run time.
7. Click **JMS Resource**, then click **New**. Configure the destinations by creating a new JMS Queue and an new JMS Topic.
8. Create a new JMS Queue with the values shown in Table 6-4. Click **Next**

Table 6-4 JMS Queue values

Parameter	Value
Name	TradeBrokerQueue
JNDI Name	jms/TradeBrokerQueue

9. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field make sure **TradeJMSServer** is selected. Click **Finish**.

10. From the JMS Resource window, create a JMS Topic with the values listed in Table 6-5. Click **Next**.

Table 6-5 JMS Topic values

Parameter	Value
Name	TradeStreamerTopic
JNDI Name	jms/TradeStreamerTopic

11. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field make sure **TradeJMSServer** is selected. Click **Finish.**
12. Create a second JMS Topic with the values listed in Table 6-6. Click **Next**.

Table 6-6 JMS Topic values

Parameter	Value
Name	TradeStatsTopic
JNDI Name	jms/TradeStatsTopic

13. Select **JMSServerSubDep** in the Subdeployment drop-down menu. In the Targets field make sure **TradeJMSServer** is selected. Click **Finish**.
14. Create two connection factories:
    - Queue connection factory
    - Topic connection factory

    Create the connection factories with the values shown in Table 6-7.

Table 6-7 Queue connection factory values

Parameter	Value
Name	QueueConnectionFactory
JNDI Name	jms/QueueConnectionFactory

15. In the Targets filed click **Advanced Targeting** button.
16. In the **Subdeployment** drop-down menu select **JMSServerSubDep** and in the Targets field make sure **TradeJMSServer** is selected. Click **Finish**.

17. Create a Topic Connection Factory by repeating steps 14 on page 111 through 16 on page 111 using the values listed in Table 6-8.

Table 6-8 Topic connection factory values

Parameter	Value
Name	TopicConnectionFactory
JNDI Name	jms/TopicConnectionFactory

Deploying the sample application and populating the database

The next step is to deploy the Trade application. To do this, you need to upload the `trade3wls.ear` through the WebLogic administration console.

1. Log in to the WebLogic administration console. From the home page select **Configure Applications** and then click **Install**.
2. Select **Upload your file(s).**
3. Click **Browse** and select the `trade3wls.ear` file. Under **Location**, make sure you select the **trade3wls.ear** radio button. Click **Next**.
4. Make sure **Install this deployment as an application** radio button is selected and click **Next**.
5. In the Optional Settings window, accept all defaults and click **Finish**

This process is quick and normally completes in less than a minute. At this point, you should see two successful messages, one for the deployment status for the EJB Modules and another for the deployment status for the web application modules.

Test the newly deployed application by accessing the following web page from your web browser:

`http://localhost:7001/trade`

To start loading the database with sample data, perform the following steps:

1. Select **Configuration** from the Trade home page.
2. Select **(Re)-populate Trade Database** to start loading the database with sample data.

This process takes a few minutes depending on the hardware configuration used. When finished, you should see 1000 quotes created and 500 users registered. On the left pane, click **Go Trade!** to browse the application.

If you click the Configuration link on the left side and click **Trade Scenario**, you can see how the application automatically navigates itself by clicking the **Reload** button in your browser.

This feature randomly selects the next action to be performed and populates all the data instead of the user having to do it by hand. This greatly simplifies performance scripts. If you want to performance test this application, you do not need to navigate all windows and enter all the data. You can point your stress tool to the URL `http://localhost:7001/trade/scenario` and configure your stress test tool to see how many hits and how many concurrent users access this URL, and then start your test.

6.5.3 Migrating the sample application

In this section, we describe the steps performed to migrate Trade. We configure the IBM WebSphere Application Server Migration Toolkit V1.1 in Rational Application Developer V7.5 to scan for WebLogic specific files and flag potential problems with Java source code. Then we build the Trade application for WebSphere. We also explain the issues and the solutions to the problems we experienced when migrating the Trade application.

Due to the size of this section, the following list details the subsections it contains:

- "Installing the Application Migration Tool V1.1" on page 113
- "Configuring and running the Application Migration Tool" on page 116
- "Fixing errors reported by Application Migration Tool" on page 122
- "Re-running the Application Migration Tool" on page 129
- "Migrating Trade Application J2EE Level" on page 143
- "Configuring WebSphere Application Server V7" on page 145
- "Configuring JMS" on page 146
- "Creating a new data source" on page 156
- "Migrating EJBs" on page 158
- "Specifying web application references" on page 161
- "Restarting WebSphere Application Server" on page 162
- "Deploying the application" on page 162
- "Testing the application" on page 163

Installing the Application Migration Tool V1.1

The IBM WebSphere Application Server Migration Toolkit V1.1 is a plugin for Rational Application Developer V7.5 that supports the migration of J2EE applications developed for competitive application servers to WebSphere Application Server. Currently, the tool supports the migration of EJB modules and web modules deployed for Oracle WebLogic Server 9.2 to WebSphere Application Server V7.

See 6.4, “Application Migration Tool overview” on page 103 for instructions on how to download and install the IBM WebSphere Application Server Migration Toolkit V1.1.

Importing the Trade EAR file

We start the migration of Trade by importing the EAR file with source code into Rational Application Developer V7. We also import a WAR file, which is a web services client that is needed at compile time by the Trade application.

Tip: The Application Migration Tool requires the presence of the source code to perform Java code analysis.

1. In the Rational Application Developer Enterprise Explorer perspective, right-click and select **Import** → **EAR file** from the context menu.
2. Click **Browse** and select the Trade EAR file. In our example we have it in `C:\Software\Migration\Trade3\trade3wls.ear`, as shown in Figure 6-2.

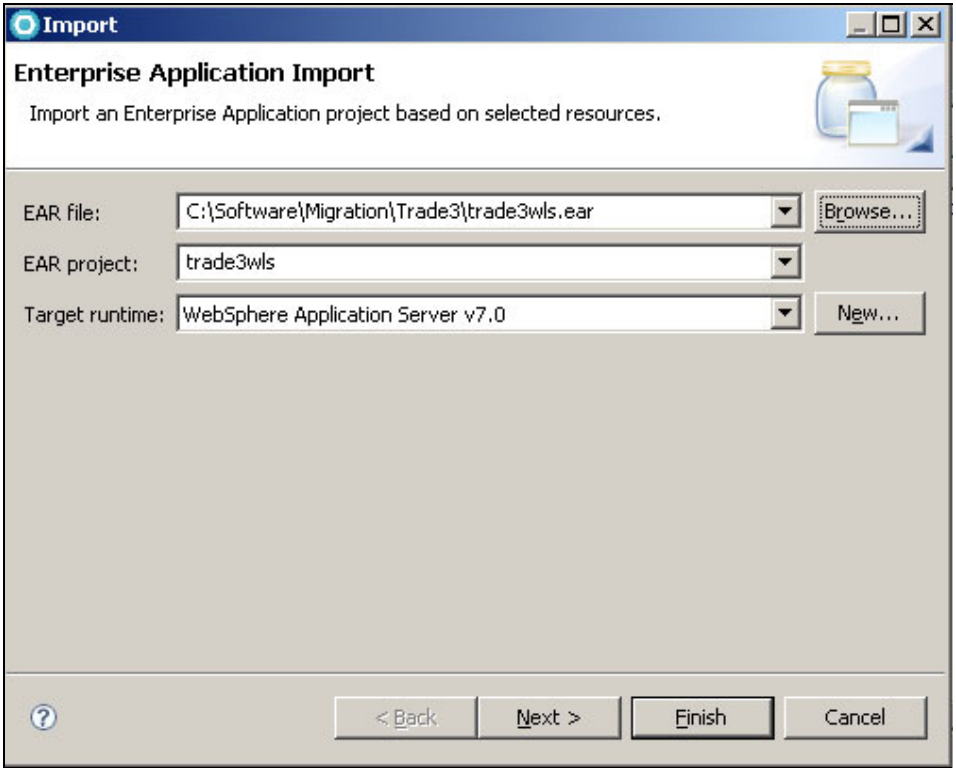


Figure 6-2 EAR file import

3. Make sure the Target runtime field is set to **WebSphere Application Server v7**.
4. In the Enterprise Application Import window, select **trade3AppClient.jar** and click **Next**.
5. In the Ear Module and Utility JAR Projects window, make sure all three modules are selected, as shown in Figure 6-3 and click **Finish**.

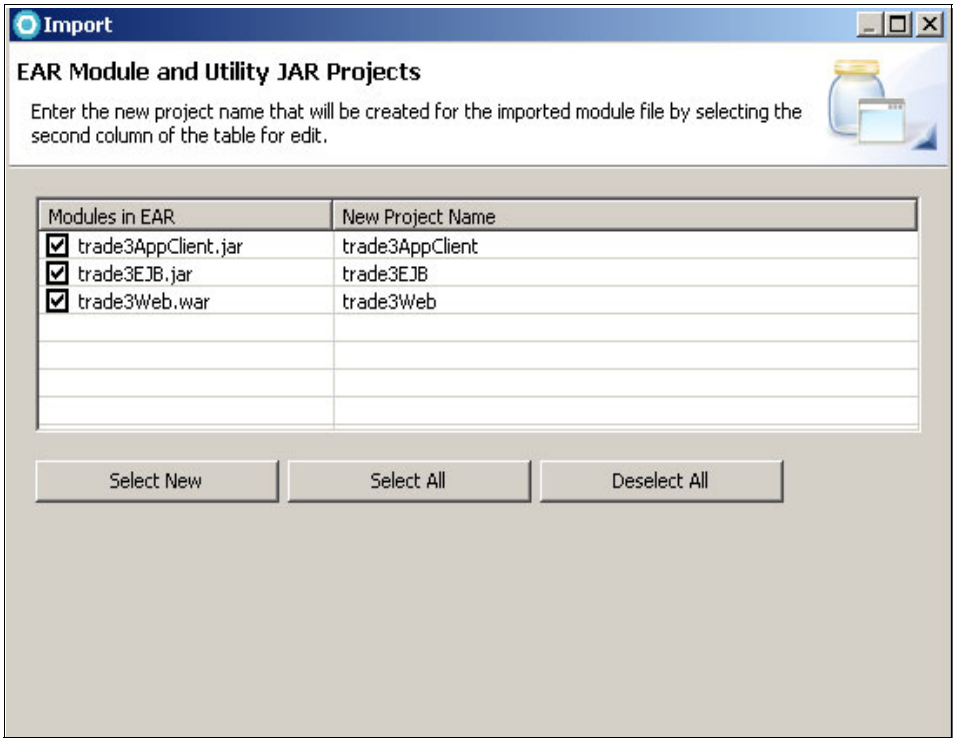


Figure 6-3 Trade module selections

It takes a couple of minutes for Rational Application Developer to build the project. You can check the status at the bottom right corner of the workbench. When the build is complete you will notice a lot of errors in the problems view. This is expected, as the application has the web logic generated classes.

Next, we import the Trade web services client WAR file.

6. In the Rational Application Developer Enterprise Explorer perspective, right-click and select **Import** → **WAR file** from the context menu.
7. Click **Browse** and select the Trade WAR file `trade3WSAppClient.war`.
8. Under EAR Membership clear the **Add project to an EAR** field.
9. Click **Finish.**

Tip: Rational Application Developer V7.5 automatically rebuilds the project after each change to source code. Turn this off by selecting **Project** → **Build automatically** from the menu.

After the import you are prompted to switch to the web perspective. Select **NO**.

You should have the following projects in the Enterprise Explorer view:

- trade3AppClient

  Contains a J2EE client application for Trade.

- trade3EJB

  This project contains the Trade EJBs.

- trade3Web

  The Trade web application containing servlets and JSPs.

- trade3

  This project contains the EAR application.

- trade3WSAppClient

  This project contains the trade web services client.

Configuring and running the Application Migration Tool

In this section we configure the Application Migration Tool analysis capabilities to scan the WebLogic Trade application files and provide an input as to what needs to be fixed in the application before it can run on WebSphere.

The Application Migration Tool needs to be set up to look for WebLogic code-related issues. However it can also be run to do a complete Java code review. The tool provides quick fixes, where available, for issues it flags as violations of the migration rules.

Important: This is the first release of the tool. It does not provide quick fix for every possible scenario. Whenever a rule violation occurs and there is a quick fix available, the rule icon includes a light bulb icon and the Quick fix option is not available in the context menu.

1. Select **Analysis** from the Run menu to display the Application Migration Tool main configuration dialog box, as shown in Figure 6-4.

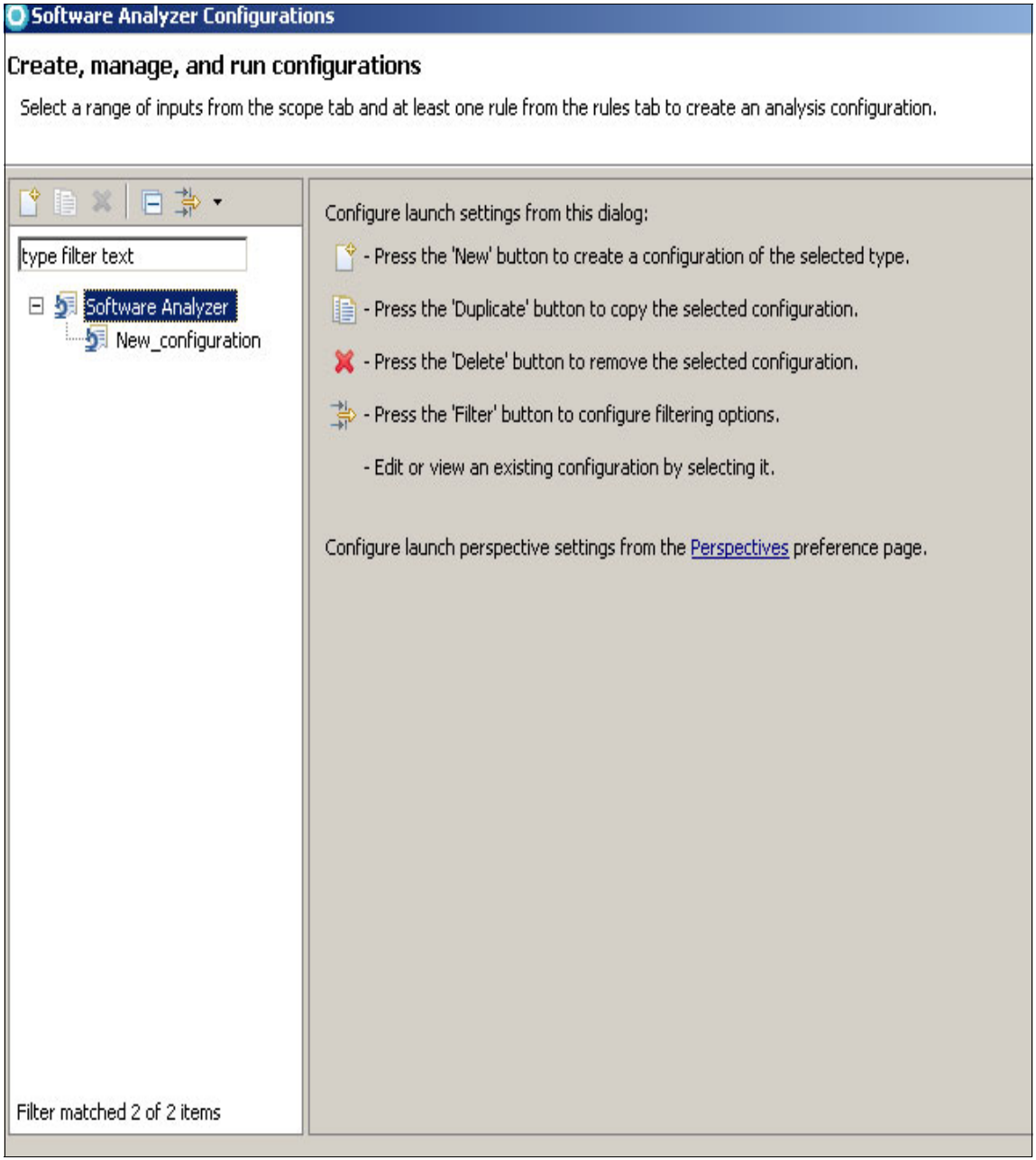


Figure 6-4 Application Migration Tool configuration

2. Expand the **Software Analyzer** and click **New_configuration** in the window shown in Figure 6-4 on page 117. The right side changes to show and explanation of the basic configuration options available.

> **Tip:** If you start a new workspace, the Software Analyzer does not have the New_Configuration sub-option. In this case highlight Software Analyzer and click the new icon on the toolbar (the first icon with the"+" sign).

3. In the Name field, enter `AppMigration` as the name of the configuration.
4. In the Scope tab, leave the **Analyze entire workspace** radio button selected, as shown in Figure 6-5.

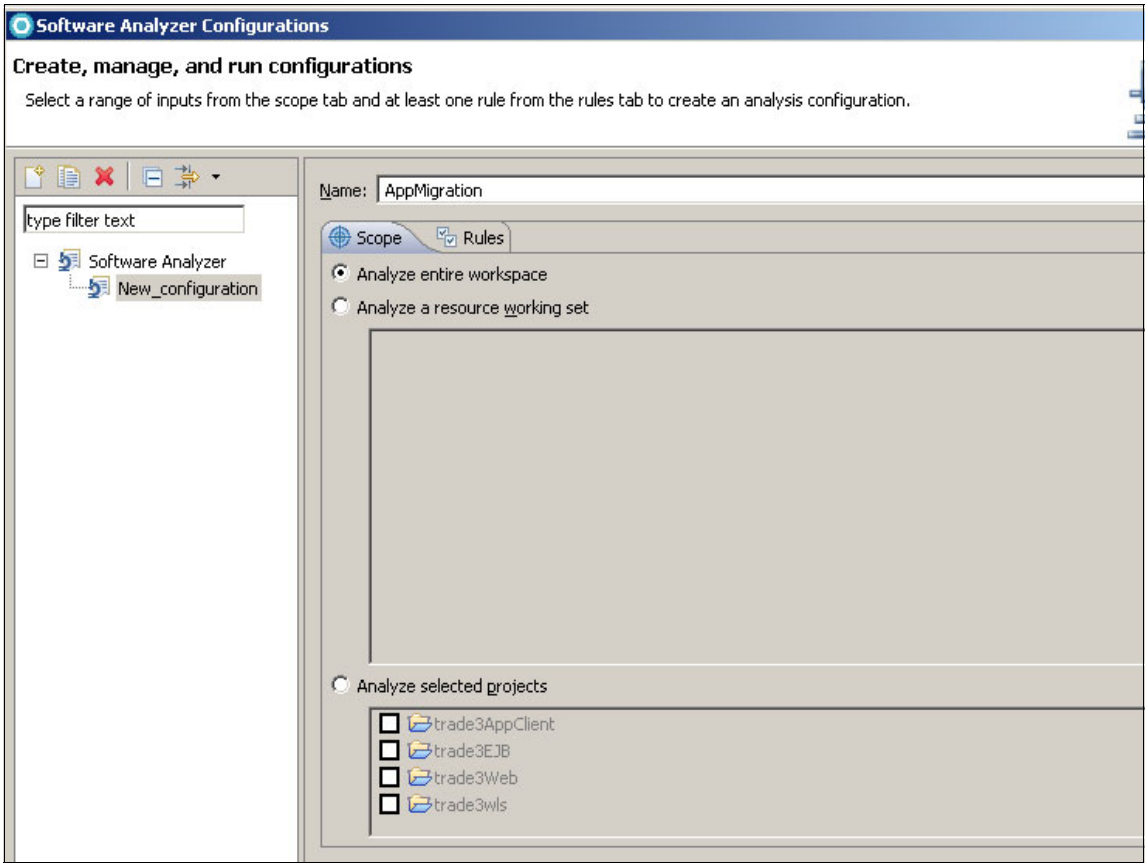


Figure 6-5 Migration tool workspace scope settings

5. Click the Rules tab.

6. Select **WebLogic Application Migration** from the Rules Sets drop-down menu, and click **Set**. See Figure 6-6.

Tip: The default is that no rules are selected until the user selects the ones that are of interest and clicks **Set** . After the configuration is set it can be reused.

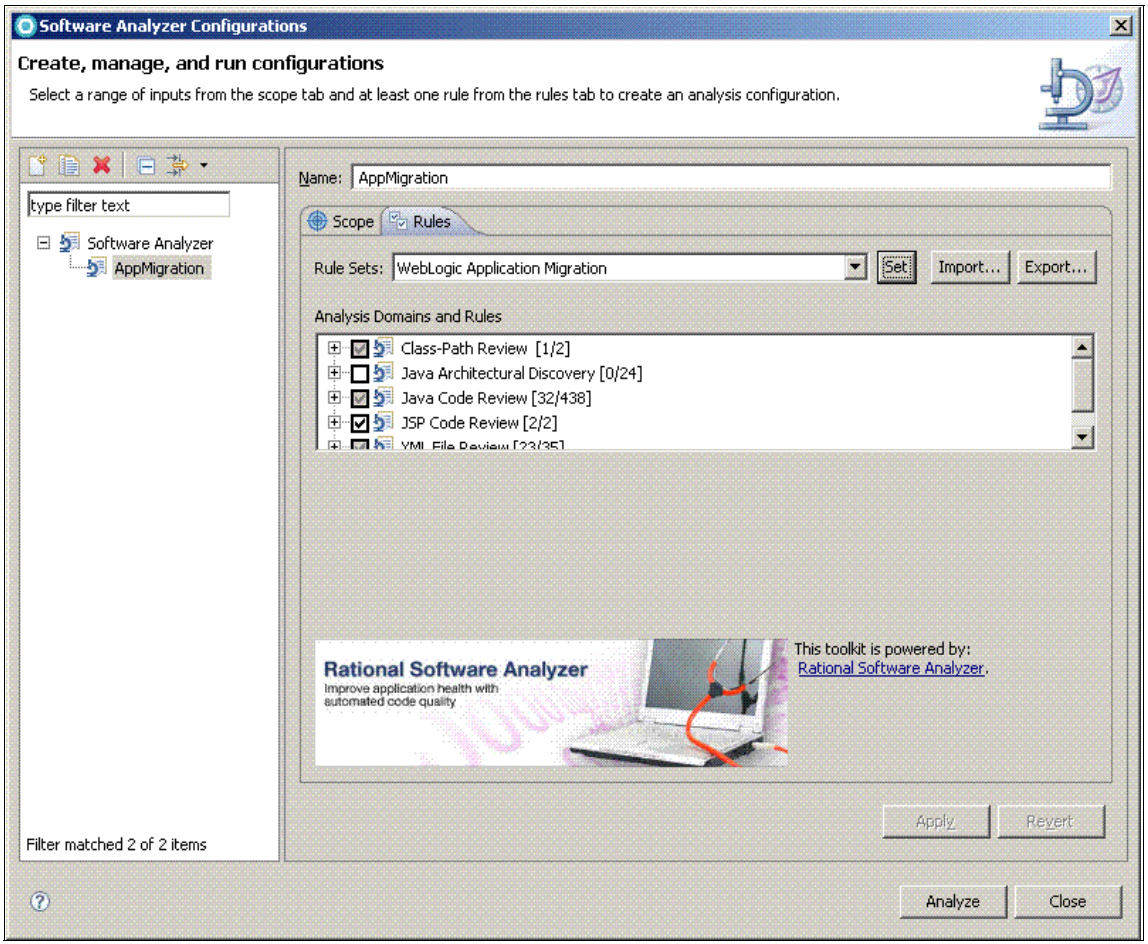


Figure 6-6 Migration Tool rule settings

The Application Migration Tool is now configured and the rules set. We are ready to run the analysis against the trade application. You can expand any of the analysis domains to see which rules are selected.

7. Click **Analyze** to analyze all the projects in the workspace.

   The Software analyzer view is opened at the bottom of the window. It contains four tabs. Each tab corresponds to one of the analysis domains selected based on the Rule Set selection made earlier.

   The Class-Path Review tab lists any class path related issues.

   The Java Code Review tab lists Java code related issues.

   The XML File Review tab lists issues with deployment descriptors and the presence of any web logic specific files.

   The JSP Code Review tab lists issues found in JSP files, commonly associated with web projects.

8. Click the XML File Review tab. See Figure 6-7.

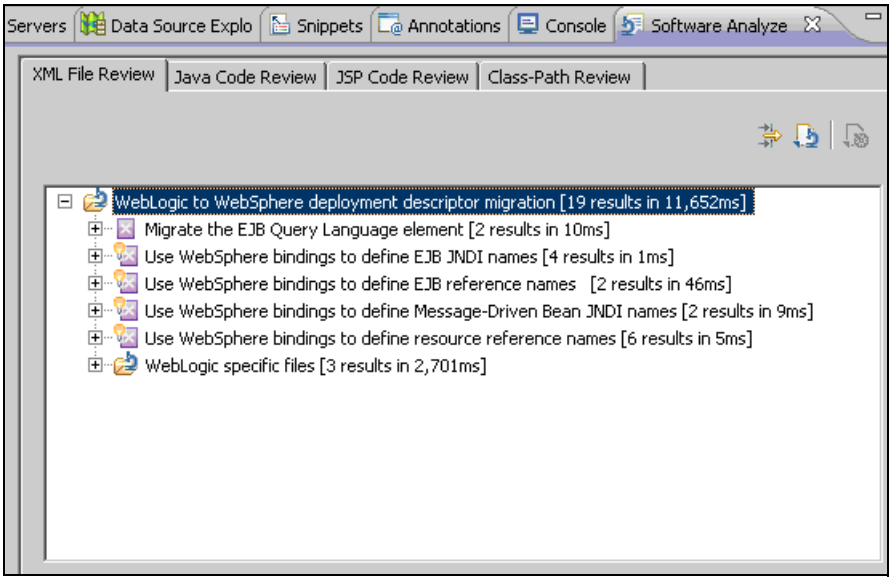


Figure 6-7 Software Analyzer View

As you can see, the tool scanned the XML files in the project and flagged the potential problems with WebLogic deployment descriptors by category/rule violation. For example, the tool points out that EJB and MDB (message-driven bean) JNDI (Java Naming and Directory Interface) references are defined in a WebSphere binding file. It also detected WebLogic specific files. You can expand each of the rules to see the files in question and which project to which they belong.

9. Click the Java Code Review tab. See Figure 6-8.

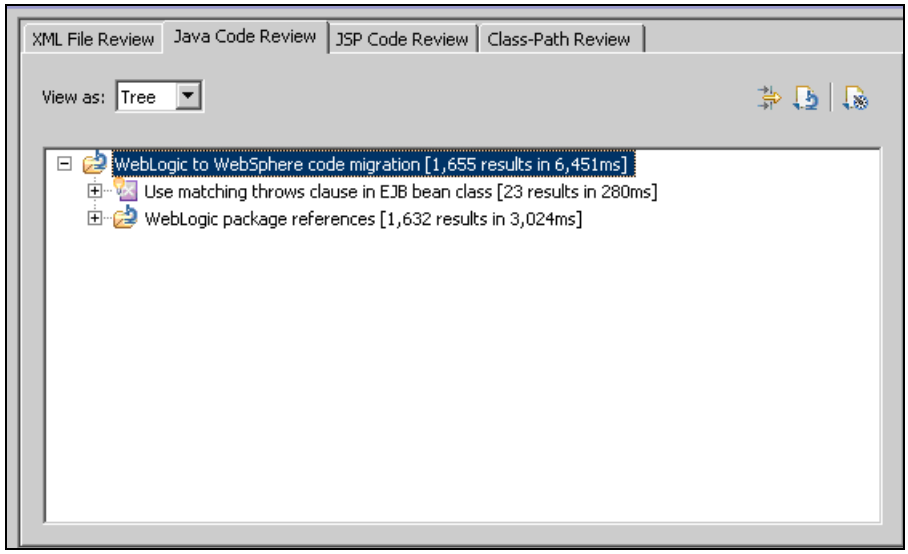


Figure 6-8 Java Code Review tab

The tool found two rule violations. The first one is the absence of using matching throw clause for Java exceptions between the EJB remote interfaces and their corresponding bean implementation class.

The tool also flagged 1632 occurrences of the use of WebLogic specific package names in Java files. Those packages were auto generated by WebLogic application server. At this point the tool only flags this issue but does not provide a quick fix option.

Tip: You can right-click any of the flagged files under a specific rule to see a list of available options in the pop-up menu. For example, under Java Code Review tab expand the WebLogic package references rule and right-click any of the files in the list. From the context menu, select **View Results**. The Java file in question is opened and the cursor is at the code problem line.

Fixing errors reported by Application Migration Tool

In this section we fix the issues reported by the Application Migration tool as and the problems reported by Rational Application Developer when we imported the Trade application.

1. Open the Problems view and analyze all the errors and warnings reported by Rational Application Developer V7.5.
2. Fix the 1632 errors reported under the Java Code Review tab with respect to WebLogic package references.
3. Expand the EJB projects and navigate to the folder containing the errors: `/trade3EJB/ejbModule/com.ibm.websphere.samples.trade.ejb`.

   This folder contains the WebLogic generated EJB source code and compiled classes. The files that contain errors are all using Oracle WebLogic Server specific classes as flagged by the Application Migration Tool.

4. Delete all the files in this folder with the “X” red mark except for the files Quote.java and QuoteHome.java.

   The com.ibm.websphere.samples.trade.ejb package resembles Figure 6-9.

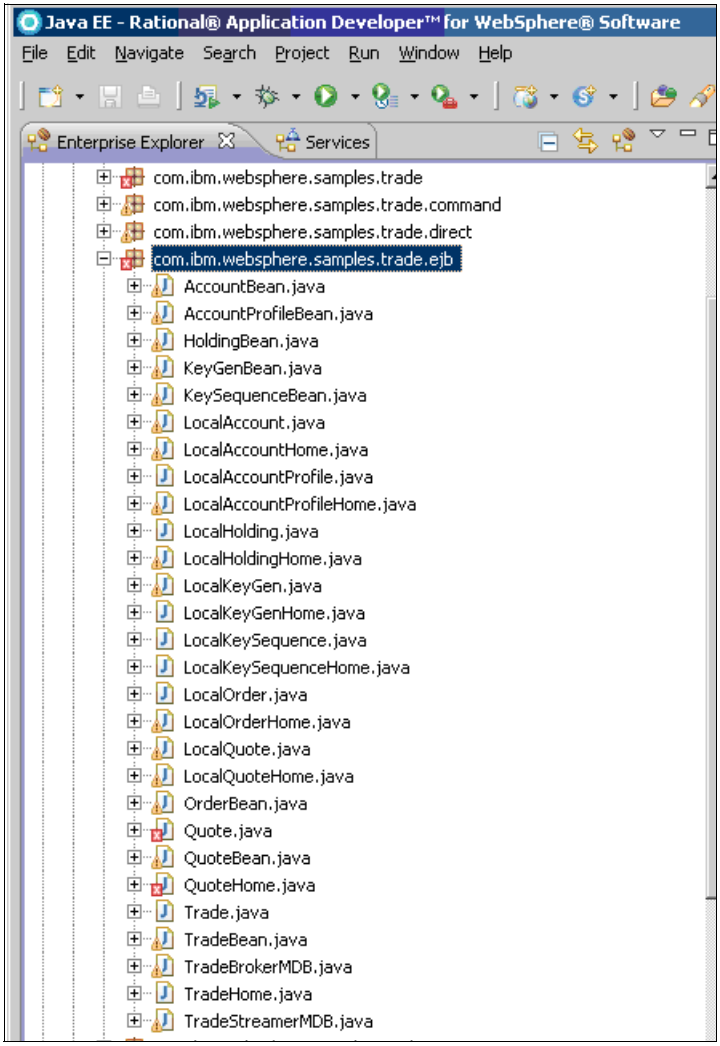


Figure 6-9 WebLogic specific packages

5. The web application project also contains Oracle WebLogic Server specific classes, so delete the `/trade3Web/Java Resources/src/jsp_servlet` folder.

   Your trade3web project should look like Figure 6-10.

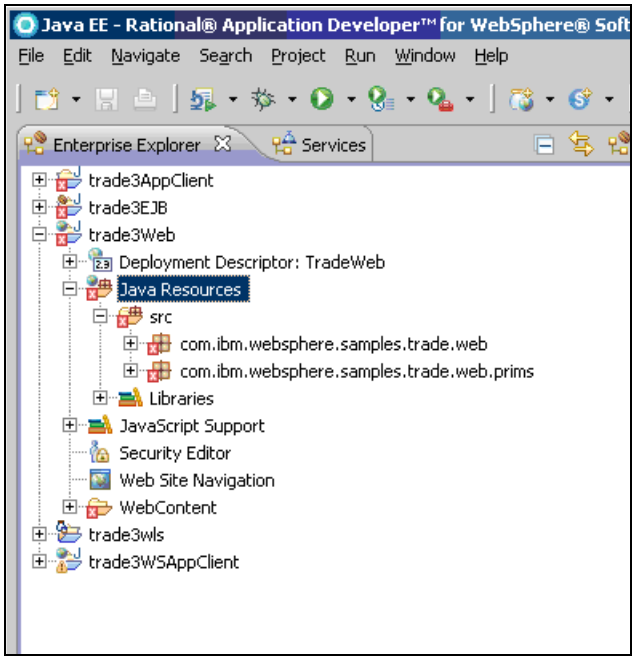


Figure 6-10 Trade web application package

Rational Application Developer V7.5 still reports errors for the web application and client project. This is because the projects depend on classes located in the EJB project. To fix the problem perform the following steps:

1. Right-click the trade3Web application project and select **Properties**.
2. From the left pane in the trade3Web properties window, select **Java Build Path**.
3. Select the Projects tab and check **Add**.
4. In the Required Project Selection dialog box select **trade3EJB** and click **OK**.

5. Click **OK** in the Properties dialog box to close it. See Figure 6-11.

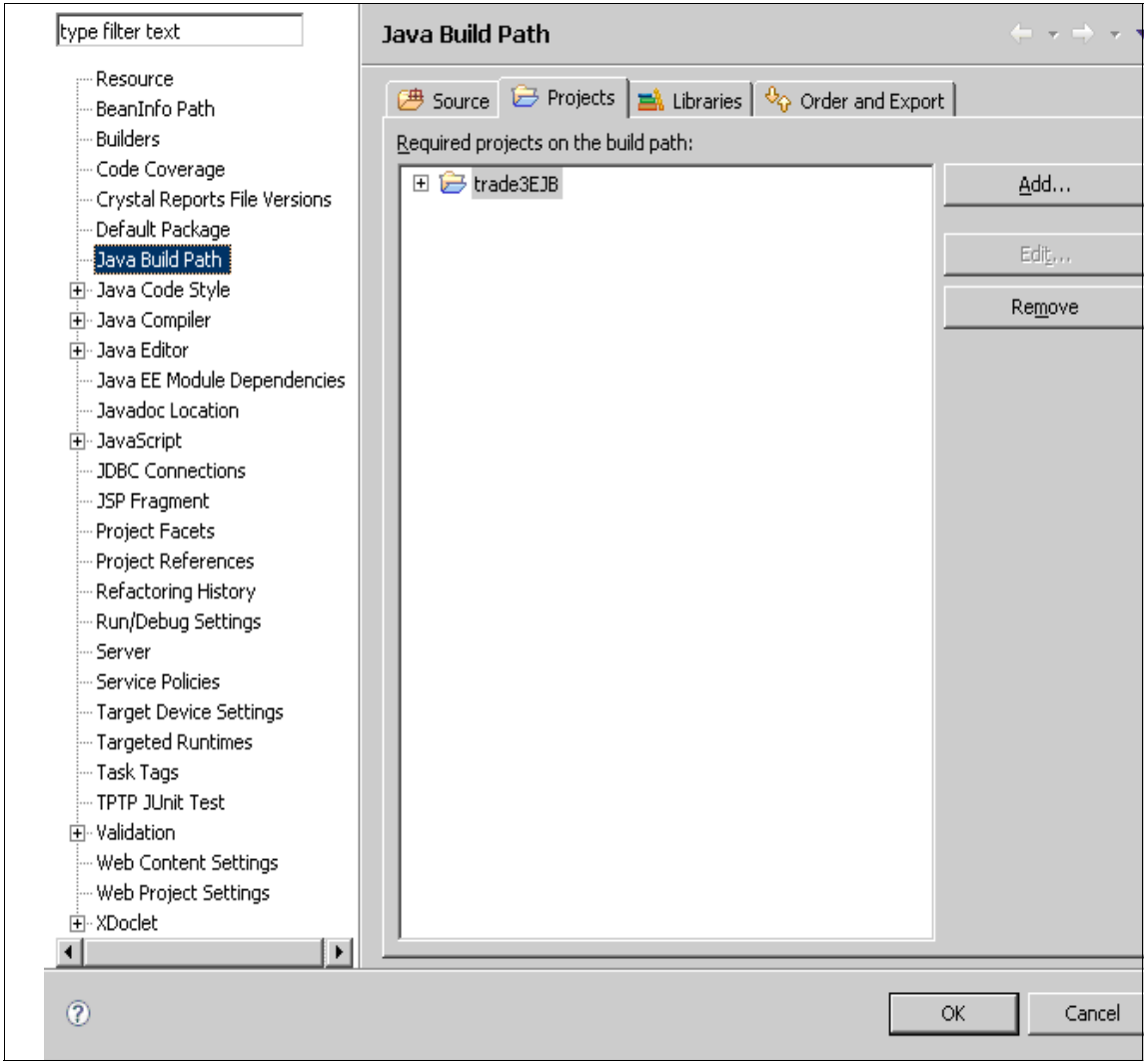


Figure 6-11 Setting Java Build Path

6. Repeat steps 1 on page 124–5 for the trade3AppClient.

   The trade3EJB project also depends on the trade web services client, so we need to set the Java build path for the trade3EJB to reference trade3WSAppClient project.

7. Repeat steps 1-5 for the trade3EJB but select **trade3WSAppClient** under the Projects tab and click **OK**.

At this point the trade3EJB project still has errors. The Problems view still shows six errors remaining for the EJB project. This is because the source code for the files Quote.java and QuoteBean.java call the Remote class in java.rmi.RemoteException package, but it does not import the package. Fix this by using the Rational Application Developer Quick Fix wizard.

8. Switch to the Problems view, as shown in Figure 6-12.

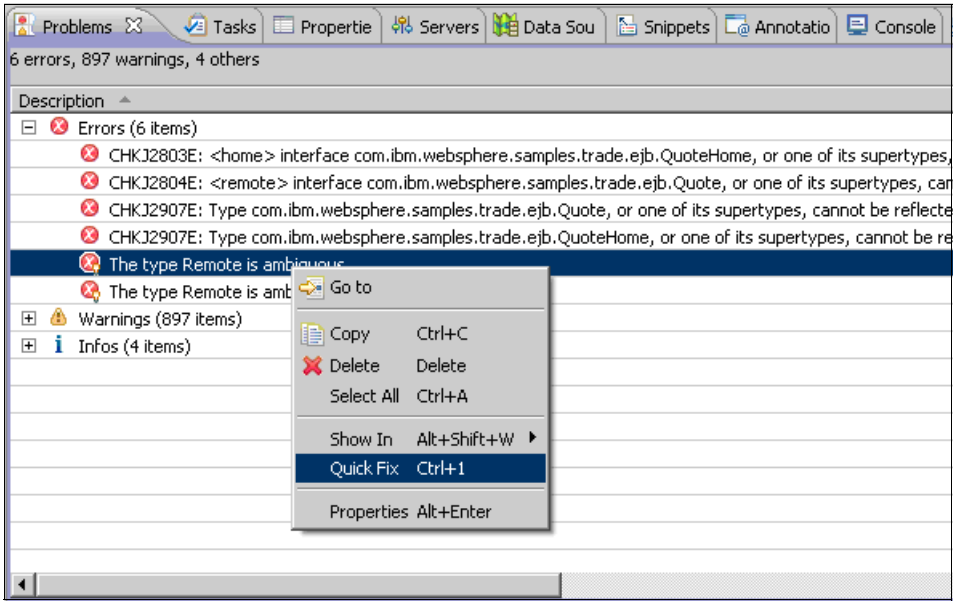


Figure 6-12 Problems View

9. Select the **The type Remote is ambiguous** error. From the context menu select **Quick Fix**.
10. The Quick Fix wizard window is displayed.
11. Highlight the first fix **Explicitly import 'java.rmi.Remote**.

12. Click **Finish**. See Figure 6-13.

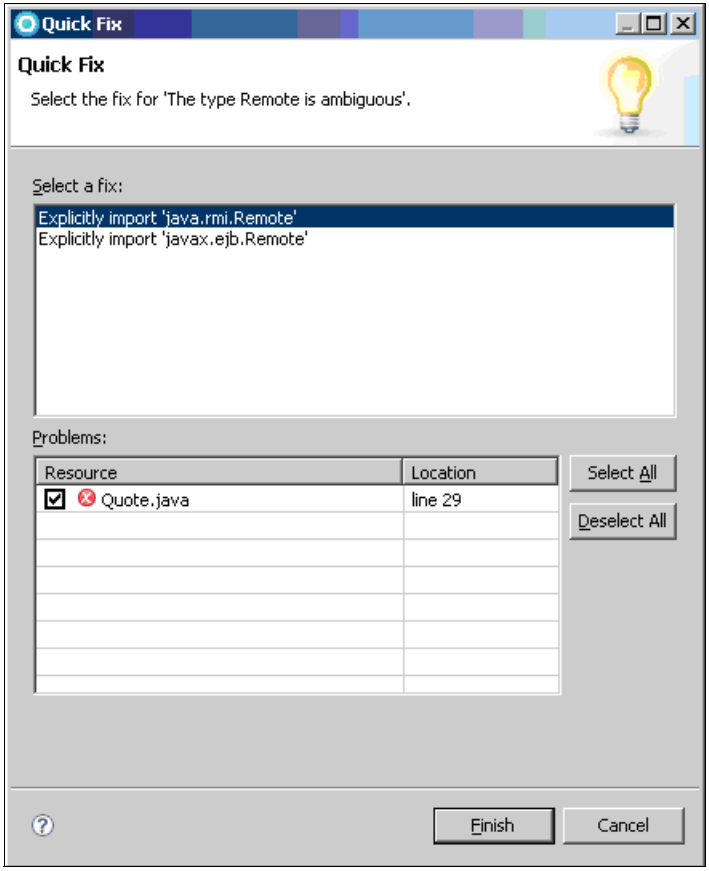


Figure 6-13 Quick Fix Wizard

The Java file `Quote.java` is open and the code modification is done by.the wizard inserting the appropriate import statement at the top of the file**.**

13. Click **Save** to save the file.

    You should only have three errors left.

14. Repeat steps 8-11 and select the **The type Remote is ambiguous** error and apply the Quick Fix as done earlier.

Important: Save the Java file after running each quick fix.

Close open Java files. The next step is to clean and rebuild all projects in the workspace.

15. From the Project menu, clear the **Build Automatically** option.
16. From the Project menu select **Clean**.
17. Select the **Clean all projects** radio button and click **OK**. See Figure 6-14.

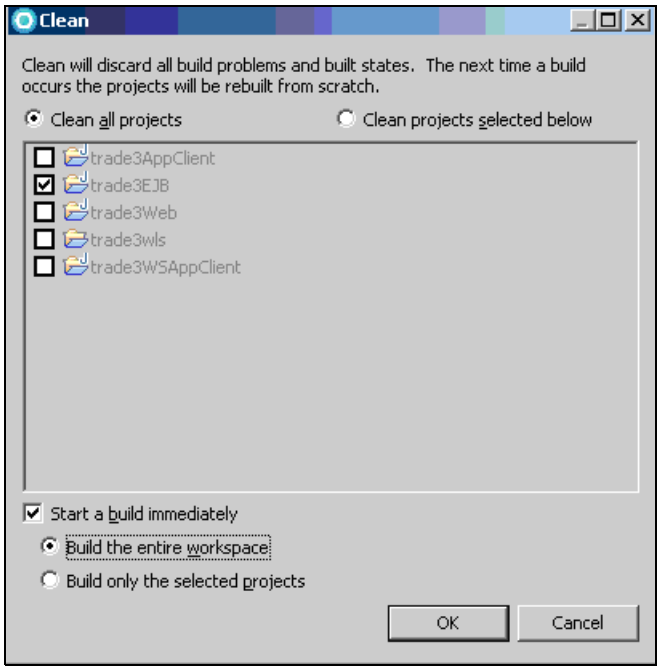


Figure 6-14 Project Clean dialog

18. From the Project menu select the **Build Automatically** option. This ensures projects are automatically built as we make changes.

All errors reported by Rational Application Developer are now fixed.

Re-running the Application Migration Tool

It is a good practice to re-run the Application Migration Tool after making a few changes to the source code. Perform the following steps:

1. From the Eclipse Run menu select **Analyze Last Launched**. See Figure 6-15.

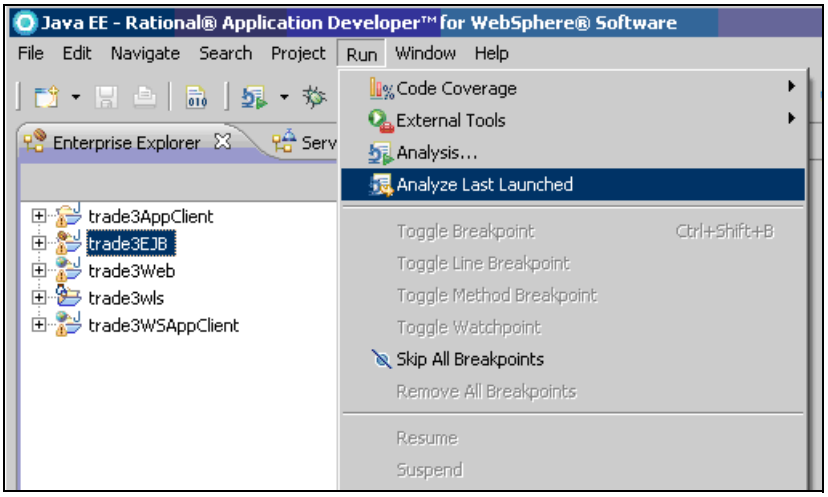


Figure 6-15 Migration tool analysis

This re-runs the analysis tool with the saved configuration. The focus is returned to the Software Analyzer Results view. Notice that the tool creates a new report in the left frame of the view with the date and time stamp.

2. Select the Java Code Review tab. You should not see the 1632 errors reported earlier by the tool.
3. Migrate the WebLogic deployment descriptors entries. The tool creates the appropriate WebSphere binding files. Click the XML File Review tab.

4. In the XML File Review tab expand the **Use WebSphere bindings to define EJB JNDI names** category. See Figure 6-16.

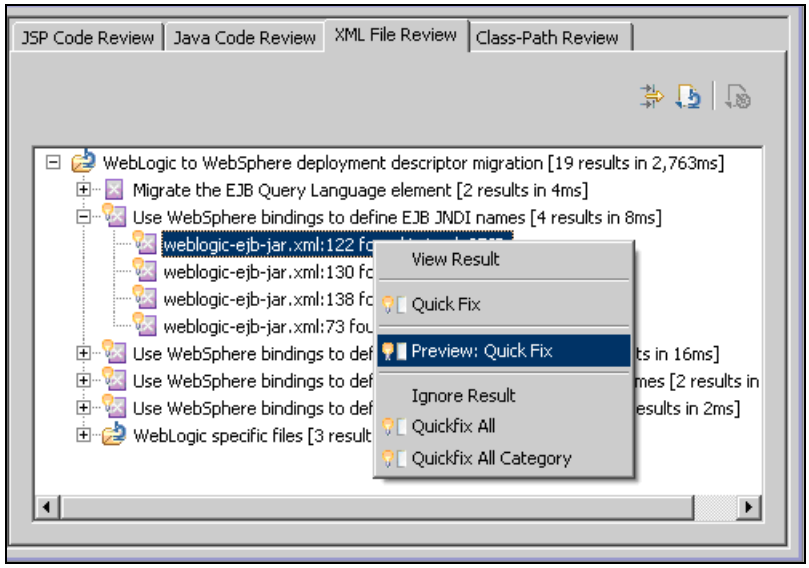


Figure 6-16 XML File Review tab

5. Right-click the weblogic-ejb-jar.xml entry. From the context menu select **Preview Quick Fix.** See Figure 6-17.

   The Quick Preview Editor tool is displayed (Figure 6-17). Examine the code changes before they are committed. In this case, the tool creates the `ibm-ejb-jar-bnd.xml` binding file and place it in the EJB module. This file did not exist because this is a WebLogic application.

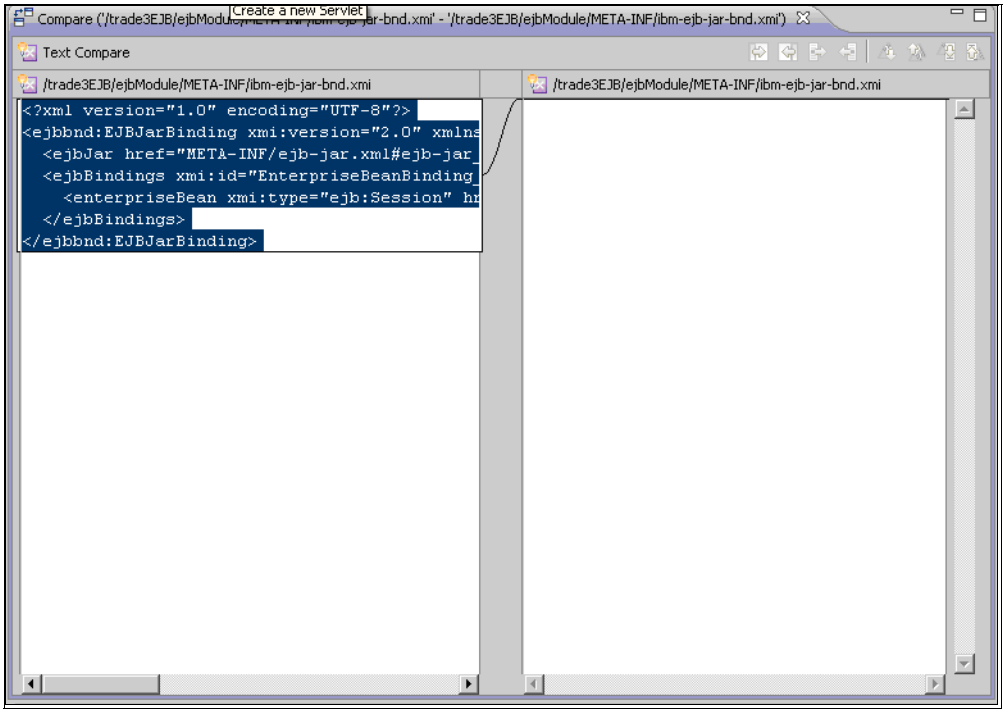


Figure 6-17 Quick Fix Preview

6. Highlight the **Use WebSphere bindings to define EJB JNDI names** category. From the context menu, select **Quickfix All**. See Figure 6-18.

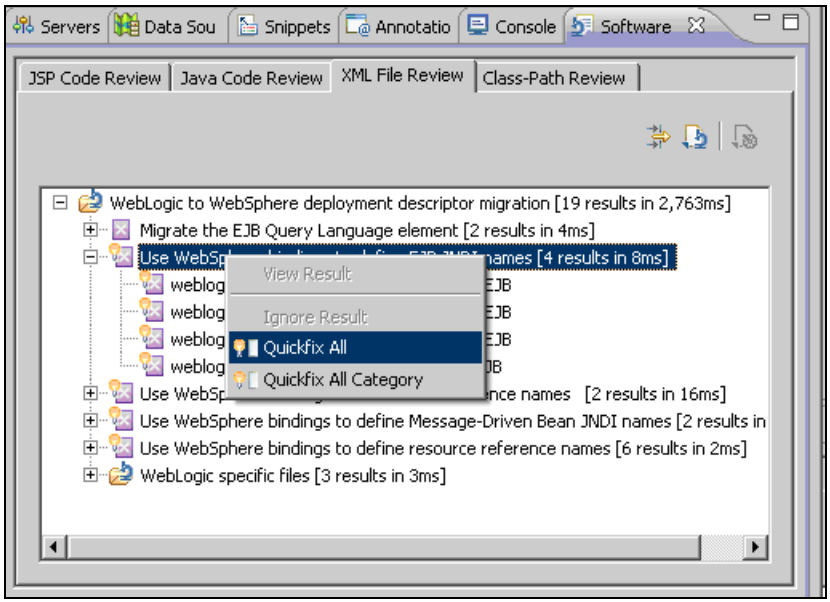


Figure 6-18 EJB JNDI Quick Fix

Important: If you select Quick FixAll category, this applies every quick fix applicable in the analysis. This is a good feature if you already know what changes the tooling will do to the application. This is not suggested for a first time application migration.

The WebSphere binding file is created and the EJB JNDI names were copied from the corresponding WebLogic deployment descriptor. Double-click the file and explore its contents. See Figure 6-19.

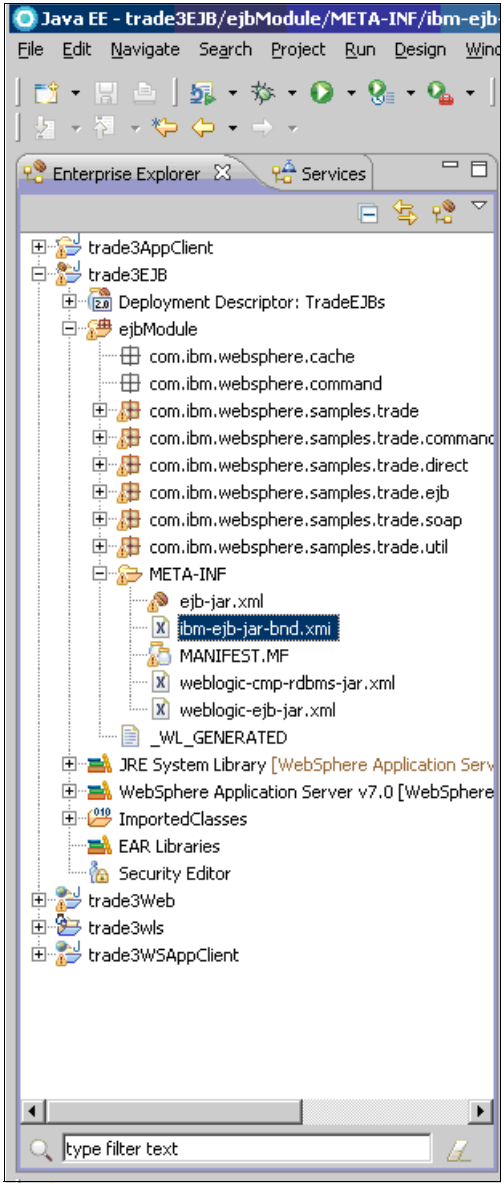


Figure 6-19 WebSphere Binding file

7. In the XML File Review tab click the **Use WebSphere binding to define EJB reference names** category. From the context menu, select **Quickfix All**.

   This rule flags EJB references found in `weblogic.xml` and `weblogic-ejb-jar.xml` if the corresponding quick fix has not already been applied. After a quick fix is applied, the WebLogic XML is marked with a comment indicating it has been migrated. This is used to determine whether to run the quick fix. It can be used at the end of migration to identify how much of the XML file has been migrated.

   The quick fix provided for this rule takes the EJB reference information defined in the WebLogic-specific deployment descriptors and migrates it to the standard deployment descriptors and the WebSphere bindings files, ibm-ejb-jar-bnd.xmi or ibm-web-bnd.xmi.

8. In the XML File Review tab highlight the **Use WebSphere bindings to define Message Driven beans JNDI names** category. From the context menu select **QuickFix All**.

   The provided quickfix copies the destination JNDI name from the `weblogic-ejb-jar.xml` to the `ibm-ejb-jar-bnd.xmi`. The quickfix, however, does not set the ActivationSpec JNDI name.

   For more information about manual intervention, highlight the rule and select the F1 function key. This launches the help section. After launching the help section, select **See details** to show the best practices for manually addressing issues such as the ActivationSpec JNDI name.

9. Repeat step 8 for the **Use WebSphere bindings to define resource reference names** category. From the context menu select **QuickFix All**.

10. Migrate your CMP beans from WebLogic to WebSphere Application Server. In this version of the migration toolkit, it defaults to an Oracle back-end. To work around this perform the following steps:

    a. Open the `weblogic-cmp-rdbms-jar.xml` file by double clicking it. This launches the deployment descriptor editor. See Figure 6-20.

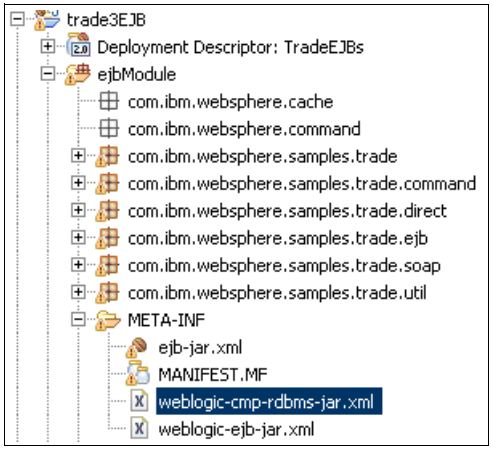


Figure 6-20 WebLogic deployment descriptor

    b. Insert the <database-type>db2</database-type> xml tag into the second-to-last field in the file. See Figure 6-21. Press Crtl + s to save this file. Otherwise, you create an Oracle back end and have to start over again.

```
                    </relationship-role-map>
                </weblogic-relationship-role>
            </weblogic-rdbms-relation>
<database-type>db2</database-type>
</weblogic-rdbms-jar>
```

Figure 6-21 Specify DB type

11. Expand the remaining rules. See Figure 6-22.

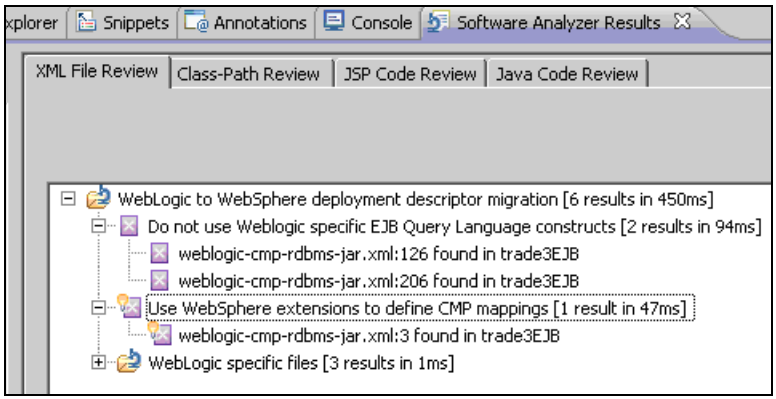


Figure 6-22 XML File Review tab

12. Review the help for the WebLogic Query language constructs by highlighting one of the rule and select the F1 function key. This launches the help section. After launching the help section, select **See details**.

13. Double-click the weblogic-cmp-rdbms-jar.xml:126 rule in trade3EJB. This shows the weblogic-ql construct. See Figure 6-23. You should see that ORDERBY is used and is a WebLogic construct and not an ejb-ql construct. The ebj-ql equivalent is ORDER BY, as shown in the help, but the weblogic-ql does not accept this. Hand-edit the `ejb-jar.xml` file after you have converted the CMP Mappings in step 14.

```
 <weblogic-query>
     <query-method>
         <method-name>findByUserID</method-name>
 <method-params>
           <method-param>java.lang.String</method-param>
         </method-params>
/query-method>
weblogic-ql>SELECT OBJECT(o) FROM Orders o WHERE o.account.profile.userID  = ?1 ORDERBY o
eblogic-query>
eblogic-rdbms-bean>
```

Figure 6-23 WebLogic Query Language construct

14. To convert the CMP Mappings, right-click the rule and select **Quick Fix**. See Figure 6-24.

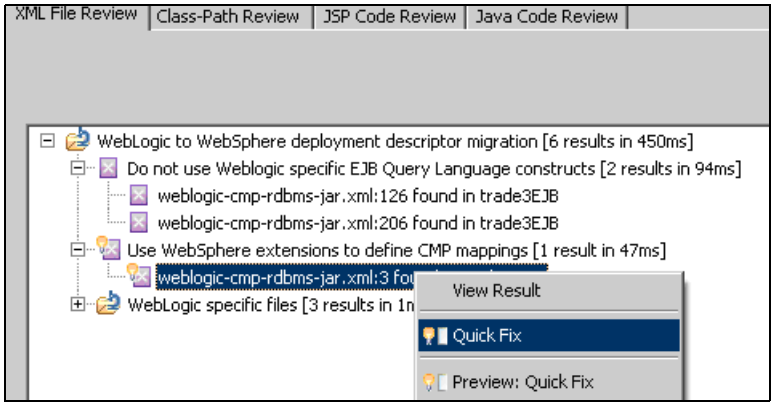


Figure 6-24 Convert CMP Mappings

15. The quick fix creates the `Map.mapxmi` file, which is used to map CMP fields to table columns.

16. Open the mapping editor to verify everything is mapped. The file is located in the following folder:

    `/trade3EJB/ejbModule/META-INF/backends/DB2UDBNT_V95_1`

    SeeFigure 6-25.

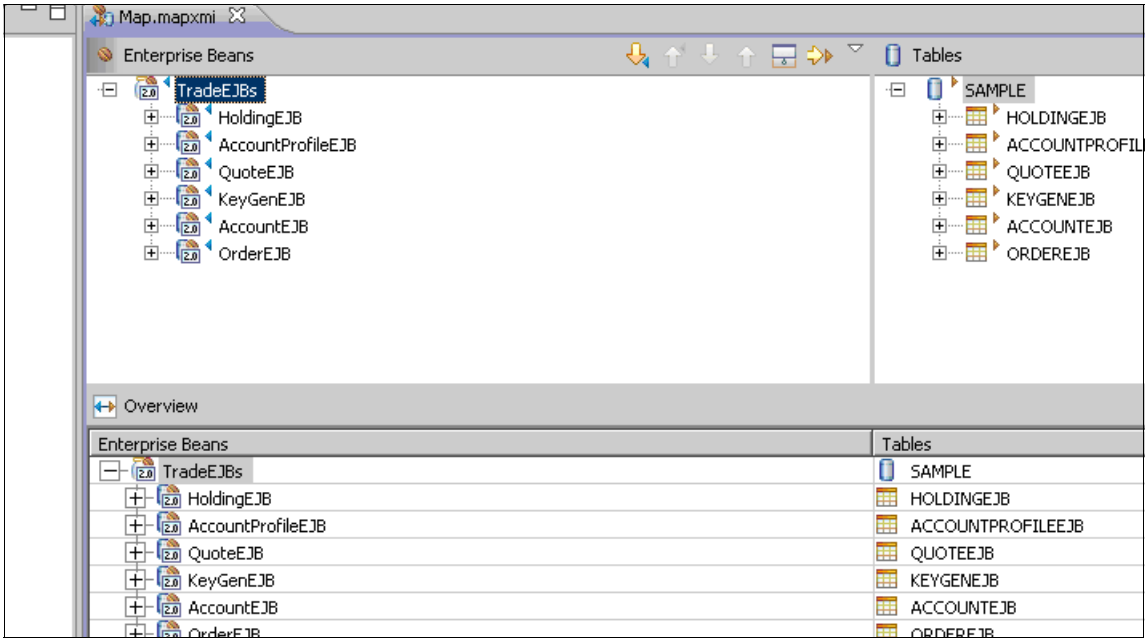


Figure 6-25 CMP napping file

The database schema is imported and located in the following folder:

`/trade3EJB/Data Models/TRADE3DB.dbm`

Figure 6-26 Database Schema files

17. To verify that all EJB elements are mapped to their corresponding table columns navigate to the following folder:

    `/trade3ejb/ejbModule/META-INF/backends/DB2UDBNT_V95_1`

    Open the `Map.mapxmi` file and enable the **Show only the unmapped objects** option. See Figure 6-27.

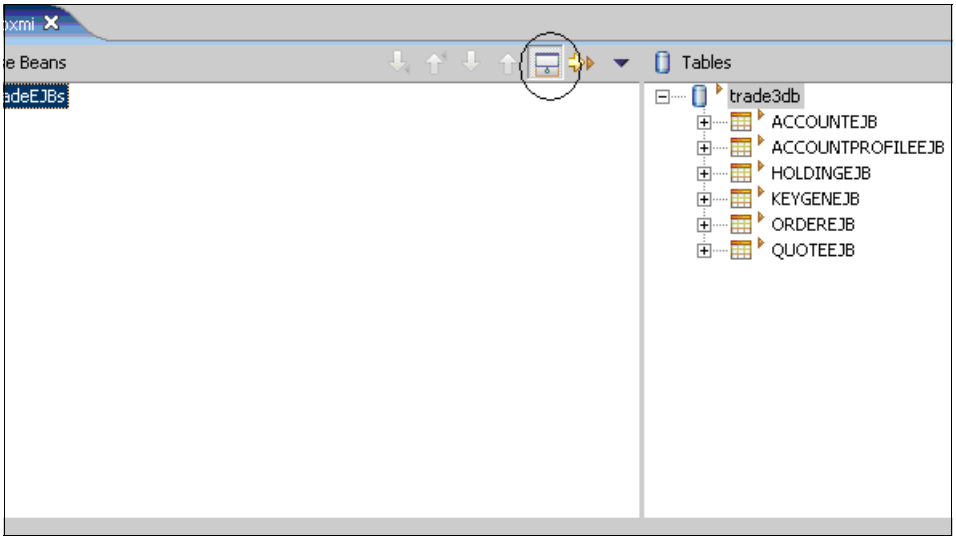


Figure 6-27 Show unmapped objects

18. Close the mapping file.

Important: Verify that the back end created is the default one. Open the `/trade3ejb/ejbModule/META-INF/ejb-jar.xml` file with the Deployment Descriptor Editor. On the Overview tab, scroll down and verify that the Backend ID is DB2UDBNT_V95_1.

19. Switch to the Source tab of the EJB deployment Descriptor. Perform the following steps to address the weblogic-ql rule pertaining to ORDERBY.

    a. Click Ctrl+F to find `SELECT OBJECT(o) FROM Orders o WHERE o.account.profile.userID = ?1`. There should only be one instance of this in the file.

    b. Replace this with `SELECT OBJECT(o) FROM Orders o WHERE o.account.profile.userID = ?1 ORDERBY o.orderID DESC`, which is from the first `<weblogic-ql>` tag in `weblogic-cmp-rdbms-jar.xml`.

    c. Change the `ORDERBY` to `ORDER BY`.

20. Repeat step 20 for the next rule. Search for `SELECT OBJECT(q) FROM Quote q WHERE q.symbol LIKE` and replace the entire ejb-ql line with `SELECT OBJECT(q) From Quote q WHERE q.symbol LIKE 's:_' OR q.symbol LIKE 's:__' ORDERBY q.change DESC`.
21. Save and close the EJB deployment descriptor editor.
22. Return to the Software Analyzer view and right-click the weblogic-ql rule. From the context menu, select **Ignore Result**. This removes the rule from the view. It also enters a comment in the `weblogic-cmp-rdbms-jar.xml` file to tell the analyzer to ignore the rule in the next iteration.
23. Run a project clean and review the Problems view. If you missed changing ORDERBY to ORDER BY, RAD will flag this as an error with a red X. Open the Source tab of the deployment descriptor and search for “ORDERBY” and make appropriate changes. See Figure 6-28.

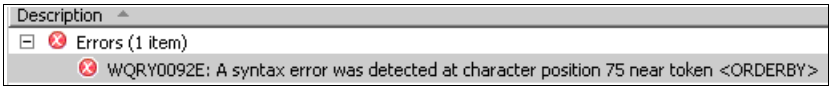


Figure 6-28 WebLogic Query Language error

24. In the XML Review tab expand the WebLogic specific files category and expand all sub-options. Those are the WebLogic specific deployment descriptors files. Delete them because we all migrated all of their entries to the corresponding WebSphere binding files.

25. In the Enterprise Explorer view expand the EJB project and browse to the `/trade3EJB/ejbModule/META-INF` folder and delete the following files:

   - `weblogic-ejb-jar.xml`
   - `weblogic-cmp-rdbms-jar.xml`

   See Figure 6-29.

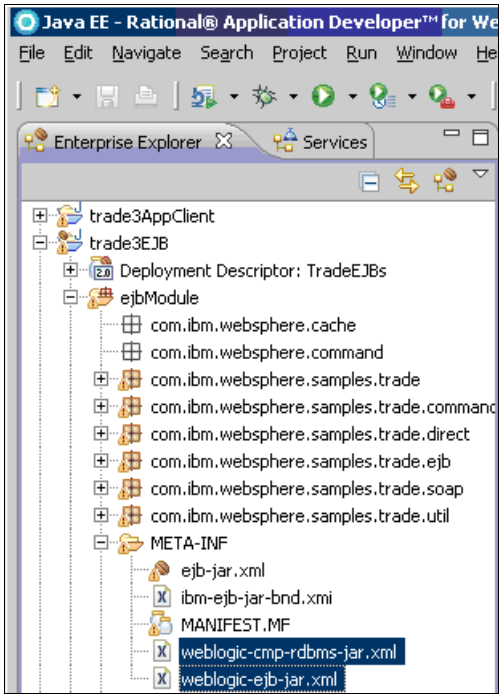


Figure 6-29 WebLogic EJB deployment descriptors

26. In the Enterprise Explorer view, expand the trade3web project and browse to the `/trade3web/webContents/WEB-INF` folder. Delete the `weblogic.xml` file. See Figure 6-30.

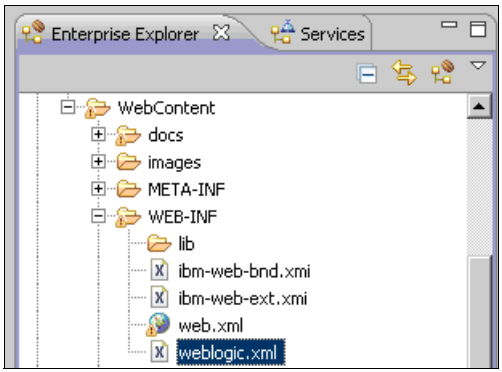


Figure 6-30 WebLogic web deployment descriptor

All errors reported by the migration tool and Rational Application Developer v7.5 should now be fixed.

Migrating Trade Application J2EE Level

Because this application is J2EE 1.3, we shall migrate this application's structure to J2EE 1.4 specification. Review each window before going to the next step.

1. In the Java EE perspective, right-click **trade3wls**. From the context menu, select **Java EE** → **Specification Migration Wizard.** See Figure 6-31.

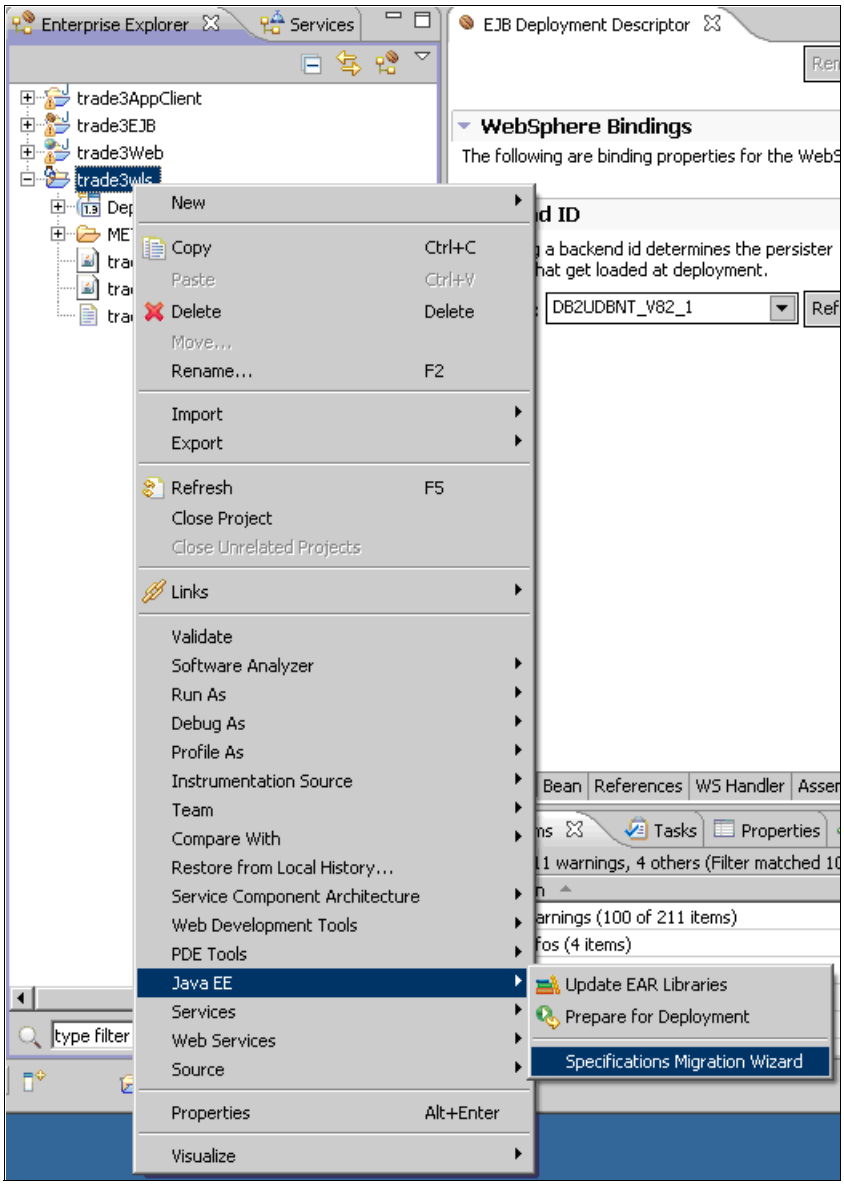


Figure 6-31 J2EE Migration Wizard

The next window provides informational messages before running the J2EE specification migration wizard. Click **Next**.

2. In the J2EE version drop-down menu, select **1.4** and click **Next**. See Figure 6-32.

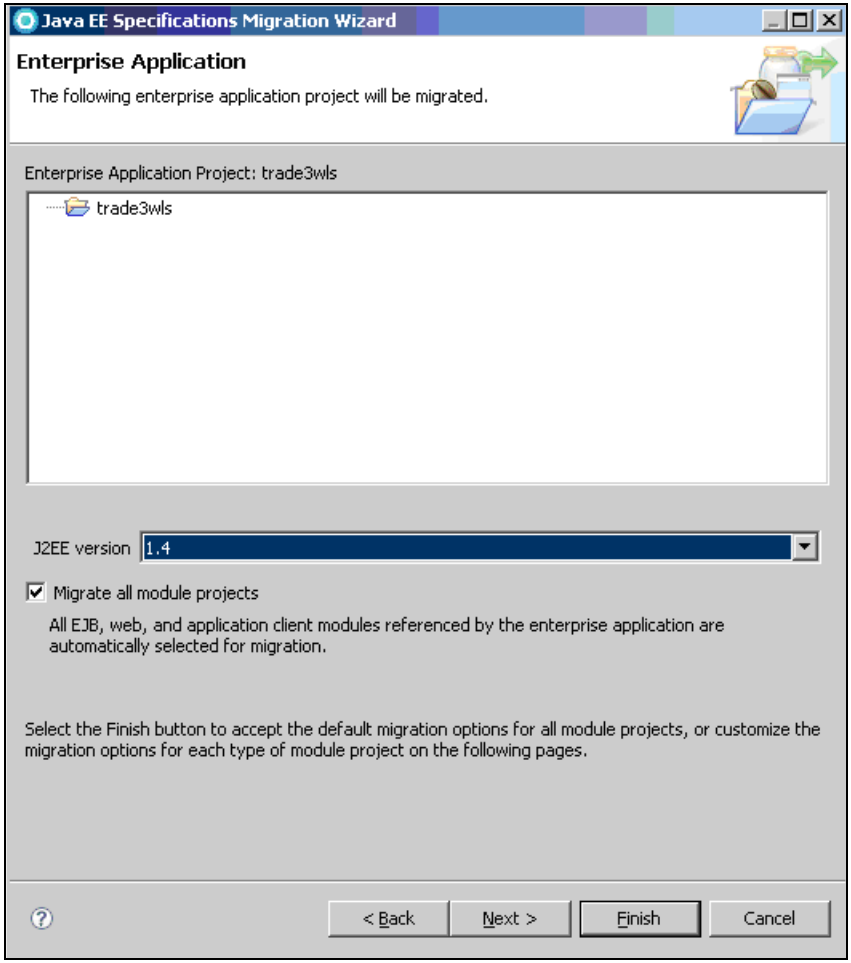


Figure 6-32 J2EE version selection

3. In the EJB Module Migration window make sure **trade3EJB** is selected and click **Next**.

4. In the Web Projects window make sure **trade3web** is selected and click **Finish**.

   The Migration Complete dialog box (Figure 6-33) is displayed.

Note: You can click the **Details** button on the Migration Complete dialog box to see the exact changes the migration wizard performed against the project.

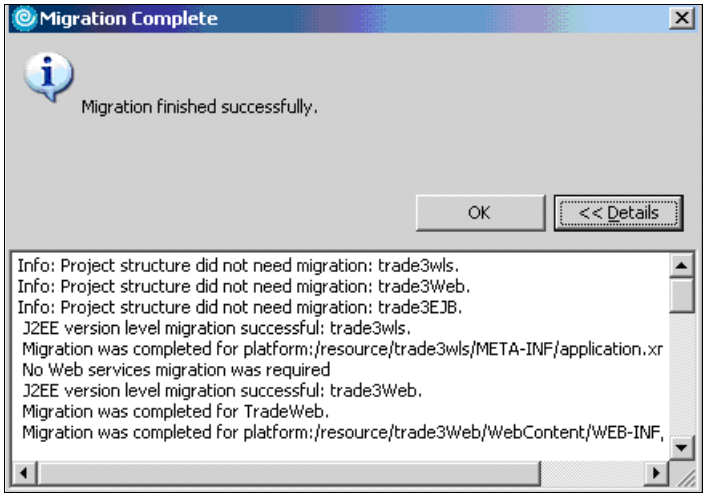


Figure 6-33 Migration Completion dialog

5. Click **OK**.

Configuring WebSphere Application Server V7

In this section, we describe how to configure WebSphere Application Server V7 to run Trade. You might want to create new WebSphere profile and a new WebSphere test server based on that profile.

Verify WebSphere Application Server is running

Verify that WebSphere Application Server is running by using the following commands:

1. In a command prompt, navigate to:

   `C:\<Rad_Install_Root>\runtimes\base_v7\profiles\AppSrv1\bin`

2. Issue the **`startServer server1`** command to start the server.
3. After a successful startup of the server, you see the message `Server server1 open for e-business`. This indicates the server is ready.

Tip: You can start the WebSphere Application Server with Rational Tool by clicking the Servers view, highlighting the server and clicking the green arrow.

Once the server is started we need to configure WebSphere Application Server V7, open a web browser and access the Administrative Console at the following URL: `http://localhost:9060/ibm/console`.

The major difference compared to Oracle WebLogic Server 9.2 is in how we configure JMS. This is because WebSphere Application Server V7 uses Service Integration Bus, which is an ESB-like approach, as the back end for the default JMS messaging engine that is based on JMS 1.1 specifications.

Note: A service integration bus in WebSphere supports applications using message-based and service-oriented architectures (SOAs). A bus is a group of one or more interconnected servers or server clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members. More information about SIB can be found in the WebSphere information center at the following web page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/concepts/cji_learning.html`

Configuring JMS

In this section, we describe how to configure the components necessary for JMS messaging. We use the browser-based administrative GUI, but this task can also be scripted using wsadmin command line tool. This way it takes only a few moments to create all necessary resources on a new server.

This task is broken into parts. Each following section describes a part of the overall process:

- “Creating a new bus” on page 147
- “Creating a bus member” on page 149
- “Creating bus destinations for JMS queues and topics” on page 149
- “Creating a JMS connection factory” on page 150
- “Creating the JMS queue” on page 152
- “Creating a second JMS queue” on page 152
- “Creating a JMS Topic” on page 153
- “Creating a JMS activation specification for the JMS queues” on page 154
- “Configuring Message Driven beans” on page 155

Creating a new bus

1. Open the WebSphere Administrative Console. Select **Service Integration** → **Buses** in the left navigation pane.
2. On the following page, create a new ESB. Specify `bus` as the name. Click **OK.**
3. Click **Finish** on the Summary window. See Figure 6-34.

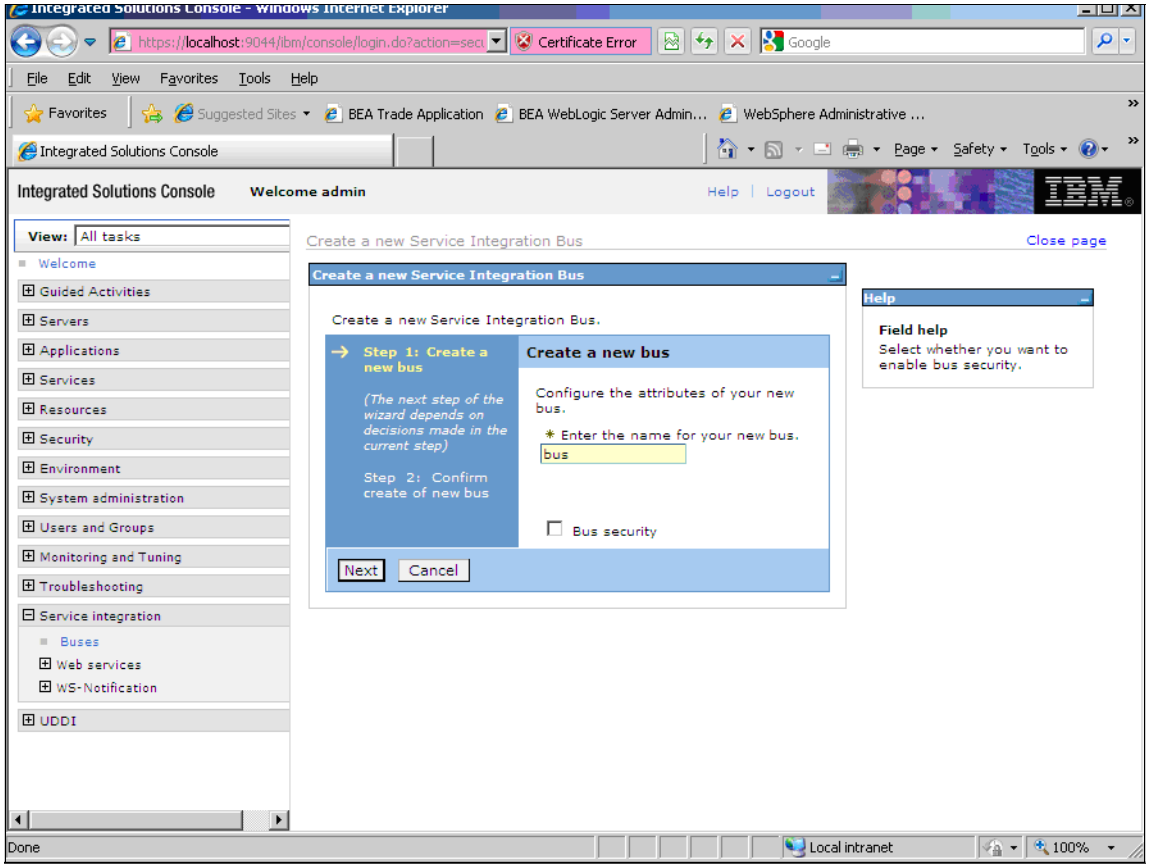


Figure 6-34 Creating Service Integration Bus

4. Save the changes made to the configuration by clicking **Save** at the top of the window. You should see the newly created bus listed under buses. See Figure 6-35.

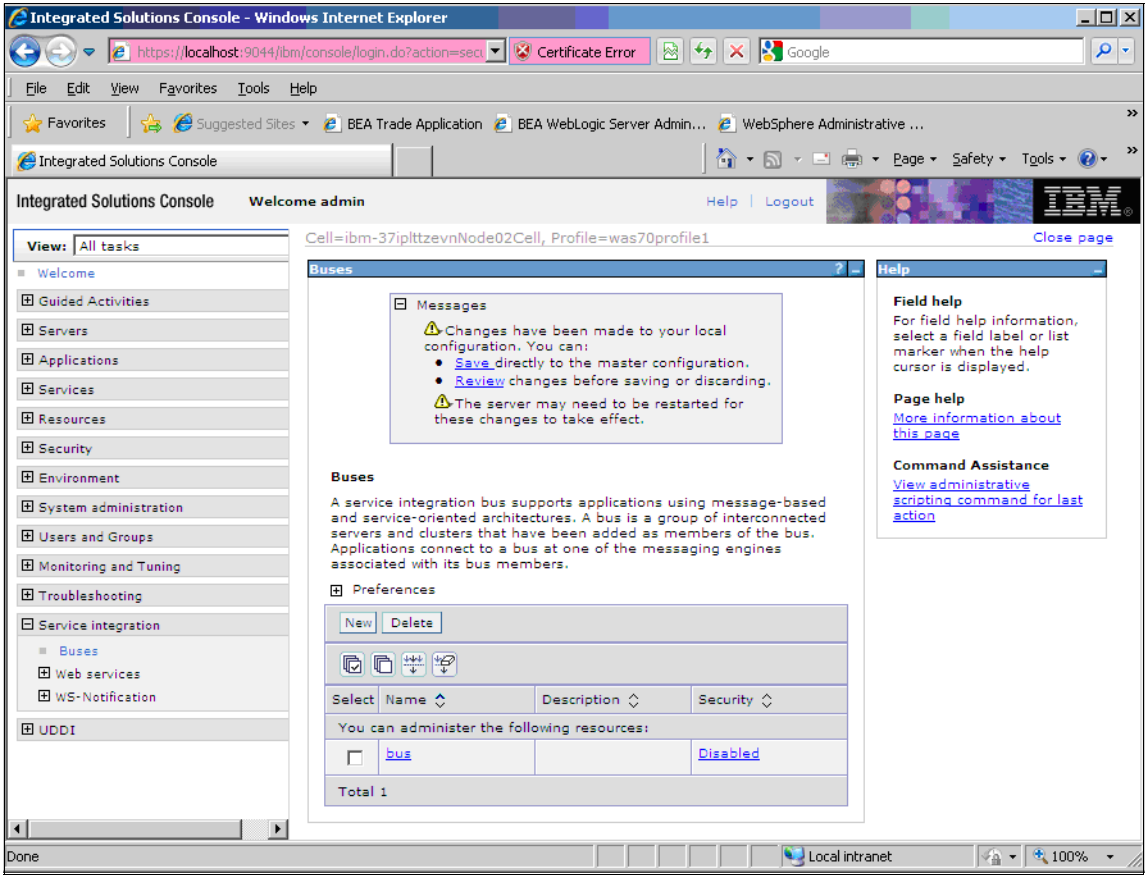


Figure 6-35 Service Integration Bus

Creating a bus member

The bus must be deployed to a bus member. The members of a service integration bus are the application servers or clusters in which messaging engines for that bus can run.

1. From the WebSphere Administrative Console, select **Service Integration** → **Buses** in the navigation pane on the left. Select the bus created in “Creating a new bus” on page 147.
2. Click **Bus members** in the Topology section.
3. Click **Add** and select the server to which the bus is deployed. Accept all defaults on the remaining windows and click **Finish**. Save the changes.

Tip: Manual configuration is error-prone and difficult. Automating these tasks saves time. Download the WebSphere version of the Trade application together with JACL scripts for the wsadmin scripting facility. Those scripts create all necessary resources and perform application deployment in a clustered environment. Download it from the following web page:

`http://www.ibm.com/software/webservers/appserv/was/performance.html`

Refer to the WebSphere Information Center for more details about the administrative scripting:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/welcadminscript.html`

Creating bus destinations for JMS queues and topics

The destination is the target for all messages sent to the enterprise service bus. Follow these steps to create the bus destinations for the JMS queue and topic:

1. From the administration console, navigate to **Service Integration** → **Buses**.
2. Select the bus created in “Creating a new bus” on page 147.
3. Click **Destinations** in the Destination resources section.
4. Click **New**, and then select **Queue** as the destination type. Click **Next**.
5. Specify **TradeBrokerJSD** as the identifier. Click **Next**.
6. Assign the new queue to a bus member. Because there is only one bus member, click **Next**, then **Finish**.
7. Create another queue destination by repeating steps 1 through 6. Use `TradeBrokerDirect` as the identifier.
8. Create a topic space in the same way by repeating steps 1 through 6. Specify `Trade.Topic.Space` as the identifier.
9. Save the changes.

Creating a JMS connection factory

Before configuring the JMS queues, you need to configure a JMS connection factory, which is used by the Trade MDBs to create a connection to the JMS provider.

1. From the administration console, navigate to **Resources** → **JMS** → **JMS providers** from the left navigation pane.
2. In the right frame select **Node=<node_name>,Server=server1** in the scope drop-down menu. See Figure 6-36.

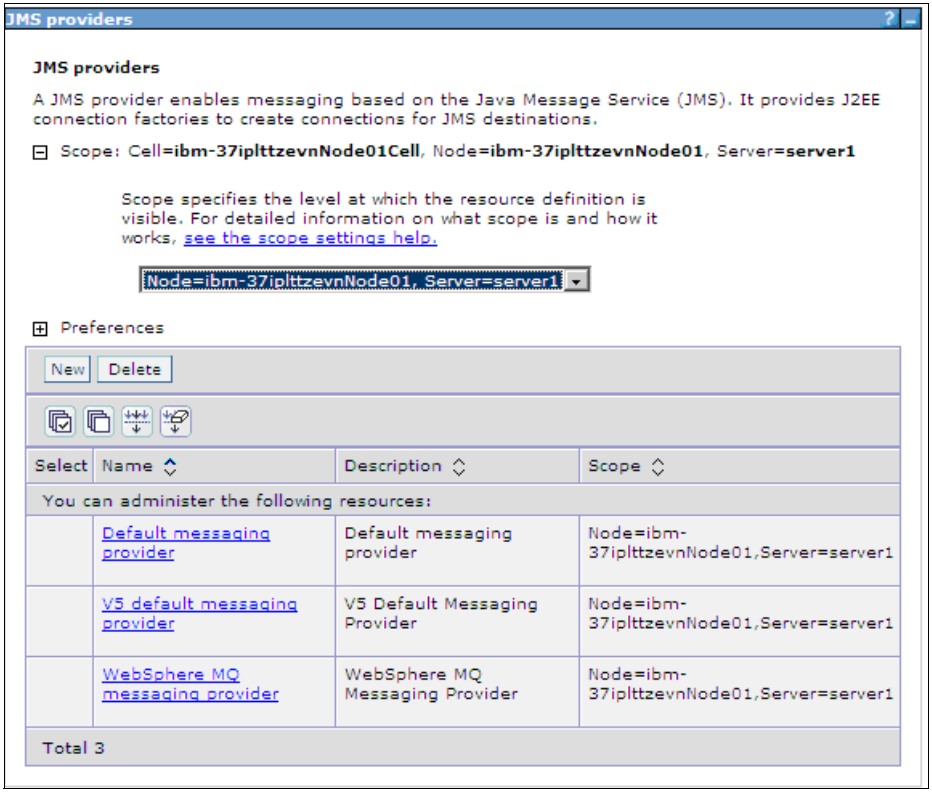


Figure 6-36 WebSphere scope selection

3. Click **Default messaging provider**.
4. Click **Queue connection factories** in the Additional Properties section.

5. Click **New**, and create a queue connection factory using the information shown in Table 6-9.

Table 6-9 JMS queue connection factory configuration

Parameter	Values
Name	TradeBrokerQCF
JNDI name	jms/TradeBrokerQCF
Bus name	bus

6. Click **OK**, then save the changes you made.
7. Return to **Resources** → **JMS** → **JMS Providers.**
8. Click **Default messaging provider**.
9. Click **Topic connection factories** in the Additional Properties section.
10. Click **New**, and create a new JMS topic connection factory using the information in Table 6-10.

Table 6-10 JMS topic connection factory configuration

Parameter	Values
Name	TradeStreamerTCF
JNDI name	jms/TradeStreamerTCF
Bus name	bus

11. Save the changes.

Creating the JMS queue

Trade contains two message driven beans (MDBs). One MDB requires that you configure a new JMS queue. Perform the following steps to do so.

1. In the right frame click **Default messaging provider**.
2. Click **Queues** in the Additional Properties section.
3. Click **New**, and create a new JMS queue using the information in Table 6-11.

Table 6-11 JMS queue configuration

Parameter	Value
Name	TradeBrokerQueue
JNDI name	jms/TradeBrokerQueue
Bus name	bus
Queue name	TradeBrokerJSD

4. Click **OK** and save your changes.

Creating a second JMS queue

The second MDB in Trade requires that you configure a new JMS topic. Perform the following steps to do so.

1. Repeat steps 1 through 3 of the procedure in “Creating the JMS queue” on page 152 to create a second JMS queue using the information shown in Table 6-12.

Table 6-12 JMS queue configuration

Parameter	Value
Name	TradeBrokerDirectQueue
JNDI name	jms/TradeBrokerDirectQueue
Bus name	bus
Queue name	TradeBrokerDirect

2. Click **OK** and save the changes.

Creating a JMS Topic

Trade contains two MDBs. The second MDB is for the JMS topic. Follow these steps for configure the JMS topic.

1. From the administration console, navigate to **Resources** → **JMS** → **JMS providers** from the navigation pane.
2. In the right frame select **Node=<node_name>,Server=server1** in the scope drop-down menu.
3. Click **Default messaging provider**.
4. Click **Topics** in the **Additional Properties** section.
5. Click **New**, and create a new JMS topic using the following information. See Table 6-13.

Table 6-13 JMS Topic configuration

Parameter	Value
Name	TradeStreamerTopic
JNDI name	jms/TradeStreamerTopic
Bus name	bus
Topic space	Trade.Topic.Space

6. Click **OK** and save the changes.

Creating a JMS activation specification for the JMS queues

The last thing to configure in WebSphere Application Server V7 related to JMS is the JMS activation specification, which binds the MDBs to the default JMS messaging provider. The JNDI name of the JMS activation specification that you create is used in the MDB's deployment descriptor to bind the MDB to the messaging queue.

1. In the right frame click **Default messaging provider**.
2. Click **Activation specification** in the Additional Properties section.
3. Click **New**, and create a new JMS activation specification using the information shown in Table 6-14.

Table 6-14 JMS activation specification configuration

Parameter	Values
Name	TradeBrokerMDB
JNDI name	eis/TradeBrokerMDB
Destination type	Queue
Destination JNDI name	jms/TradeBrokerQueue
Bus name	bus

4. Create a second JMS activation specification with the information shown in Table 6-15.

Table 6-15 JMS activation specification configuration

Parameter	Value
Name	TradeStreamerMDB
JNDI name	eis/TradeStreamerMDB
Destination type	Topic
Destination JNDI name	jms/TradeStreamerTopic
Bus	bus

5. Click **OK** and save your changes.

At this point, you have all the necessary components for JMS messaging configured. The next step is to configure the MDBs to use the JMS activation specification and the queues.

Configuring Message Driven beans

Use Rational Application Developer V7.5 and its Deployment Descriptor Editor to edit the standard ejb-jar.xml deployment descriptor and to create the WebSphere Application Server V7 specific deployment descriptors.

1. In Rational Application Developer open the Deployment Descriptor Editor by navigating to EJB Projects `/trade3EJB/ejbModule/META-INF` and double-clicking the `ejb-jar.xml` file.
2. Select the bean tab.
3. Select **TradeBrokerMDB** from the list of EJBs. In the WebSphere Bindings section, select **JCA Adapter** and specify the values shown in Table 6-16.

Table 6-16 OrderProcessor MDB's WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/tradeBrokerMDB
Destination JNDI name	jms/TradeBrokerQueue

You have now configured the MDB to use the JCA Activation Specification and Destination configured in the previous section. See Figure 6-37.

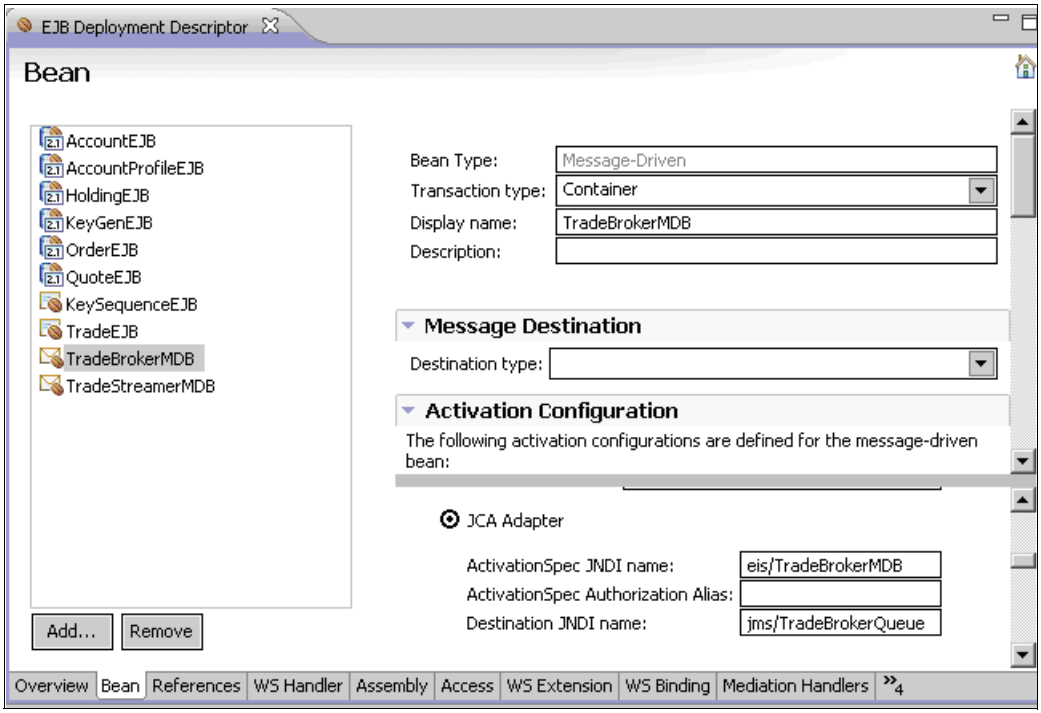


Figure 6-37 MDB Activation Spec Configuration

4. Select **TradeSteamerMDB** from the list of EJBs. In the WebSphere Bindings section, select **JCA Adapter** and specify the values shown in Table 6-17.

Table 6-17 Mailer MDB's WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/tradeStreamerMDB
Destination JNDI name	jms/TradeStreamerTopic

You have now completed the configuration of the MDBs and JMS messaging.

Creating a new data source

Trade uses a JDBC data source for connecting to the database. To set up the data source, you need to configure a JDBC provider, a J2C alias for storing the database authentication information and the data source itself.

Note: J2EE Connector Architecture (JCA) data entries are used by resource adapters and JDBC data sources. A JCA data entry contains authentication data, which includes an alias (an identifier), user ID (a user identity), password and description.

The configuration is done from the WebSphere administration console.

1. From the administration console, select **Environment** → **WebSphere Variables.**
2. In the right frame select **Node=<node_name>** in the scope drop-down menu.
3. Click the **DB2UNIVERSAL_JDBC_DRIVER_PATHI** environment variable**.**
4. Specify the correct path to the DB2 JDBC driver and click **OK**. In our migration example, the path to the JDBC driver is: `C:\IBM\SQLLIB\java`.
5. Save the changes**.**
6. Click the **UNIVERSAL_JDBC_DRIVER_PATH** environment variable and specify the value `${WAS_INSTALL_ROOT}/universalDriver/lib`**.** This points to the `db2jcc_license_cu.jar`, which is the DB2 license file.
7. Click **OK** and save the changes.

   Next, we create the JDBC provider, data source and J2C authentication alias.
8. From the administration console, navigate to **Resources** → **JDBC** → **DBC providers**.
9. In the Scope drop-down menu, select **Node=<Node-name**>.

10. Click **New** to create a new JDBC provider using the information shown in Table 6-18.

Table 6-18 JDBC provider configuration

Parameter	Value
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	XA datasource

11. We already defined the class path values for the database using WebSphere Variables, the values are already populated, so click **Next.**
12. The Summary window is displayed, click **Finis**h and save the changes.

With the JDBC provider defined and the DB2 JDBC driver path specified, the next step is to create the actual data source and authentication alias.

1. From the administration console, navigate to **Resources** → **JDBC** → **JDBC providers**, and select the newly created JDBC provider by clicking its name: **DB2 Universal JDBC Driver Provider (XA)**.
2. Click **Data sources** in the Additional properties section.
3. Click **New**, and create a new data source using the information shown in Table 6-19.

Table 6-19 JDBC data source configuration

Parameter	Value
Driver type	4
Server name	localhost
Database name	trade3db

4. Click **Next.**
5. On the Setup security alias window leave all defaults, then click **Next.**
6. The Summary scre.en is displayed click **Finish** and save the changes.
7. Select the newly created data source by clicking its name: `TradeDataSource`.
8. Click the **JAAS – J2C authentication data** in the Related Items section**.**

9. Click **New**, and create a new alias using the information shown in Table 6-20.

Table 6-20 J2C Authentication Alias Configuration

Parameter	Value
Alias	TradeDataSourceAuthData
UserId	db2admin
Password	db2adm1n

10. Click **Apply**.
11. Click **OK** and go back to the data source just created. From the Component-managed authentication alias drop-down menu, select the J2C alias just created.
12. Click **OK** and save your configuration.
13. Select the data source you just created from the list and click **Test connection**.

Migrating EJBs

This section describes how to configure the Trade Session beans and Container Managed beans to run on WebSphere Application Server V7. In our migration example, we used the Deployment Descriptor Editor in Rational Application Developer V7.5 and the migration tool to generate the WebSphere Application Server V7 specific deployment descriptors.

Specifying EJB JNDI names

The first step is to assign JNDI names to the Trade EJBs, because the migration tool did not do it for all of the EJBs.

Important: The migration tool migrated the JNDI names that were in the WebLogic EJB deployment descriptor to the corresponding WebSphere files. However, most of the EJB's JNDI names for Trade application were not specified in the EJB deployment descriptor, so we have to perform those steps manually as described.

1. From the Rational workspace, open the `ejb-jar.xml` with the Deployment Descriptor Editor and click the bean tab.
2. Select **AccountEJB**. In the WebSphere Bindings section, enter the information in Table 6-21.

Table 6-21 WebSphere bindings for the Account EJB

Parameter	Value
AccountEJB	ejb/AccountEJB

3. Repeat step two for each EJB in the list, except for the MDBs. Specify a unique JNDI name for each EJB in the following format `ejb/<EJBName>`. See Table 6-22.

Table 6-22 Trade EJB JNDI Names

EJB	JNDI Name Values
AccountProfileEJB	ejb/AccountProfileEJB
HoldingEJB	ejb/HoldingEJB
KeyGenEJB	ejb/KeyGenEJB
OrderEJB	ejb/OrderEJB
QuoteEJB	ejb/QuoteEJB
KeySequenceEJB	ejb/KeySequenceEJB
TradeEJB	ejb/TradeEJB

4. Save the changes.

Specifying EJB references

This step specifies the EJB reference names under the WebSphere Bindings for the EJBs.

5. In Rational Application Developer open the EJB deployment descriptor Editor by navigating to the `/trade3EJB/ejbModule/META-INF` folder. Double-clicking the `ejb-jar.xml` filer.
6. Select the References tab.
7. Expand AccountEJB by clicking the **+** sign next to it.
8. Select ejb/AccountProfile.
9. Under the WebSphere Bindings section, enter `ejb/AccountProfileEJB` as the JNDI reference name.
10. Repeat steps 5 through 9 using the information in Table 6-23.

Table 6-23 EJB reference names

EJB to expand	Reference	JNDI name
AccountEJB	ejb/AccountProfile	ejb/AccountProfileEJB
KeySequenceEJB	ejb/KeyGen	ejb/KeyGenEJB
TradeEJB	ejb/Quote	ejb/QuoteEJB
TradeEJB	ejb/Account	ejb/AccountEJB
TradeEJB	ejb/Holding	ejb/HoldingEJB
TradeEJB	ejb/Order	ejb/OrderEJB
TradeEJB	ejb/KeySequence	ejb/KeySequenceEJB
TradeEJB	ejb/AccountProfile	ejb/AccountProfileEJB
TradeEJB	jdbc/TradeDataSource	jdbc/TradeDataSource
TradeEJB	jms/QueueConnectionFactory	jms/TradeBrokerQCF
TradeEJB	jms/TopicConnectionFactory	jms/TradeStreamerTCF
TradeEJB	jms/TradeBrokerQueue	jms/TradeBrokerQueue
TradeEJB	jms/TradeStreamerTopic	jms/TradeStreamerTopic
TradeBrokerMDB	ejb/Trade	ejb/TradeEJB

11. Save changes to the deployment descriptor by pressing Ctrl+S. Or, from the RAD Window menu click **File** → **Save.** Do not close the EJB deployment descriptor editor. Proceed to “Configuring the default data source for CMPs” on page 161.

Configuring the default data source for CMPs

In this procedure, we specify a default data source that is used by all CMP beans.

1. Select the Overview tab in the Deployment Descriptor Editor.
2. In the WebSphere Bindings section, under the **JNDI - CMP Connection Factory Binding** heading, enter the values shown in Table 6-24.

Table 6-24 Default data source configuration

Parameter	Value
JNDI name	jdbc/TradeDataSource

Specifying web application references

Perform the following steps to specify web application references.

1. In Rational Application Developer, open the Web Deployment Descriptor Editor by navigating to the `/trade3Web/webContent/WEB-INF` folder. Double click the `web.xml` file.
2. Select the References tab.
3. Select the **jdbc/TradeDataSource** reference
4. Under the WebSphere Bindings section, enter `jdbc/TradeDataSource` as the JNDI name.
5. Repeat steps 1 on page 162 through 4 using the information in Table 6-25 for the reference and JNDI names.

Table 6-25 Trade web application reference JNDI names

Reference	JNDI name
jdbc/TradeDataSource	jdbc/TradeDataSource
jms/QueueConnectionFactory	jms/TradeBrokerQCF
jms/TopicConnectionFactory	jms/TradeStreamerTCF
ejb/Trade	ejb/TradeEJB
ejb/Quote	ejb/QuoteEJB
jms/TradeBrokerQueue	jms/TradeBrokerQueue
jms/TradeStreamerTopic	jms/TradeStreamerTopic
ejb/LocalQuote	ejb/QuoteEJB
ejb/LocalAccountHome	ejb/AccountEJB
jms/TradeBrokerDirectQueue	jms/TradeBrokerDirectQueue

Restarting WebSphere Application Server

In this procedure we restart the Restart WebSphere Application Server:

1. In a command prompt navigate to `C:\<Rad_Install_Root>\runtimes\base_v7\profiles\AppSrv1\bin`.
2. Issue the **`stopServer server1`** command to stop the server.
3. Issue the **`startServer server1`** command to start the server.

After successful startup of the server you should see the message `Server server1 open for e-business`, which indicates the server is ready.

Deploying the application

Once the resources for the application have been configured, the next step is to deploy the application to the test environment in Rational Application Developer.

1. In the Rational Application Developer Servers view, right-click **WebSphere Application Server v7.0 at localhost**. From the context menu, select **Add and Remove Projects**. See Figure 6-38. The Add and Remove Projects dialog box is displayed.

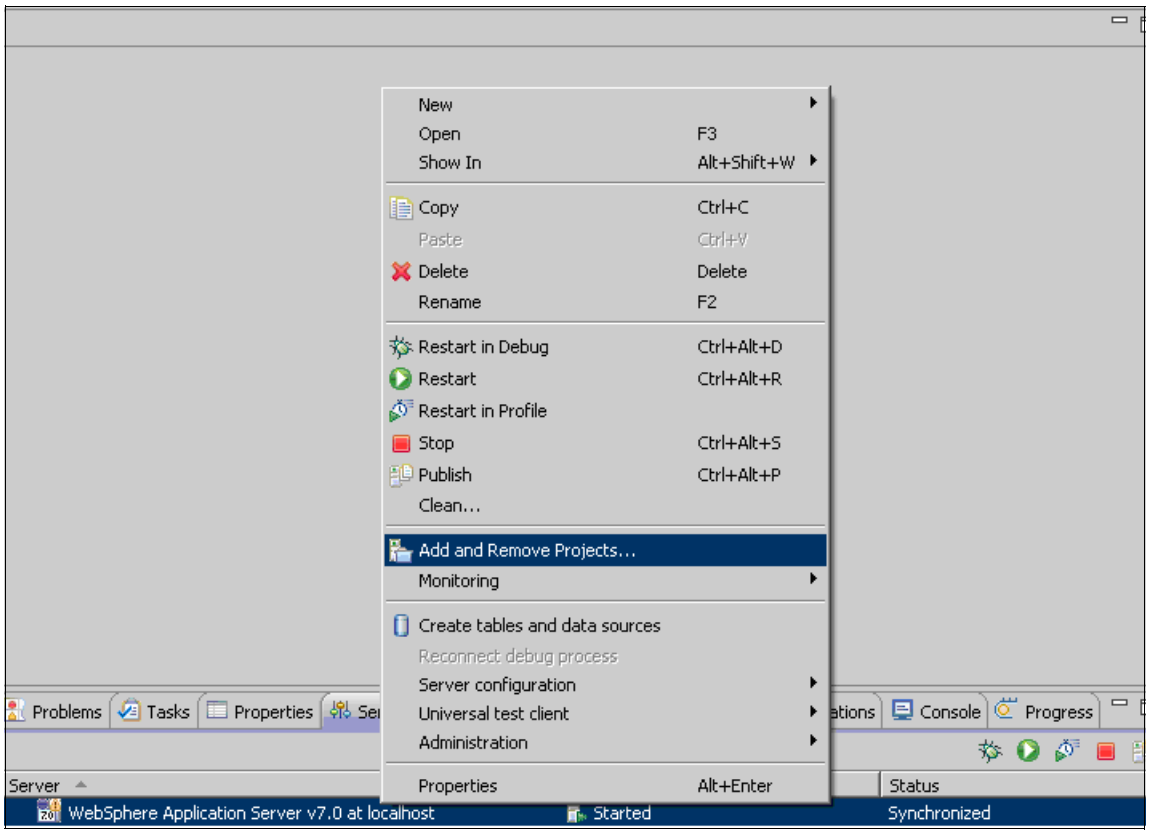


Figure 6-38 Add and Remove Projects

2. Under Available projects, select **trade3wls** and click **Add**.

   The trade3wls project is under the Configured projects.

3. Click **Finish**.

It will take a couple of minutes for the application to deploy to the server. You can expand the server and verify the application status is Started.

Testing the application

The application can be accessed at the following URL:

```
http://localhost:9080/trade
```

Figure 6-39 shows the Trade Application welcome page.

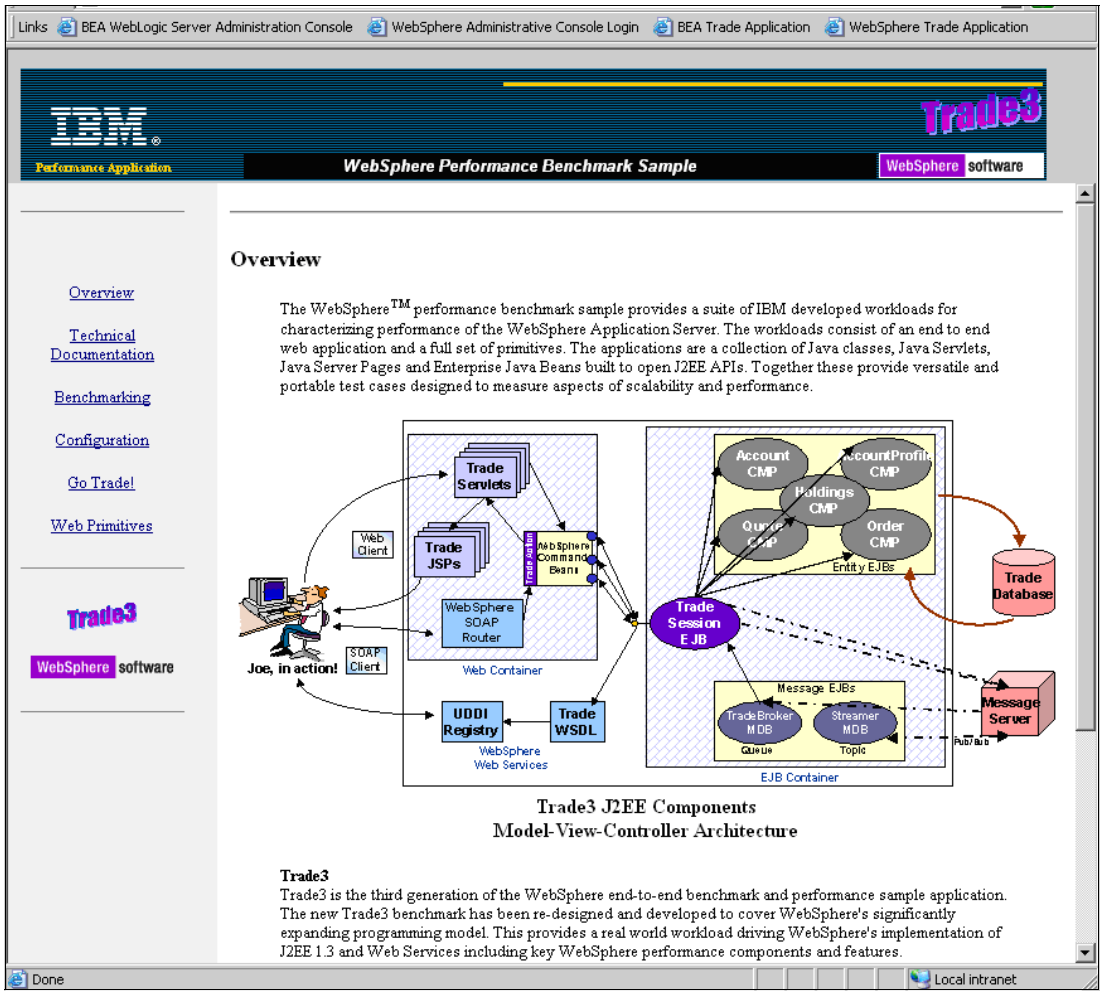


Figure 6-39 Trade Application welcome page

Browse through the site to check that the application is running correctly.

Perform the following steps to test the application:

1. Click the **Configuration** option in the left navigation pane. See Figure 6-40.

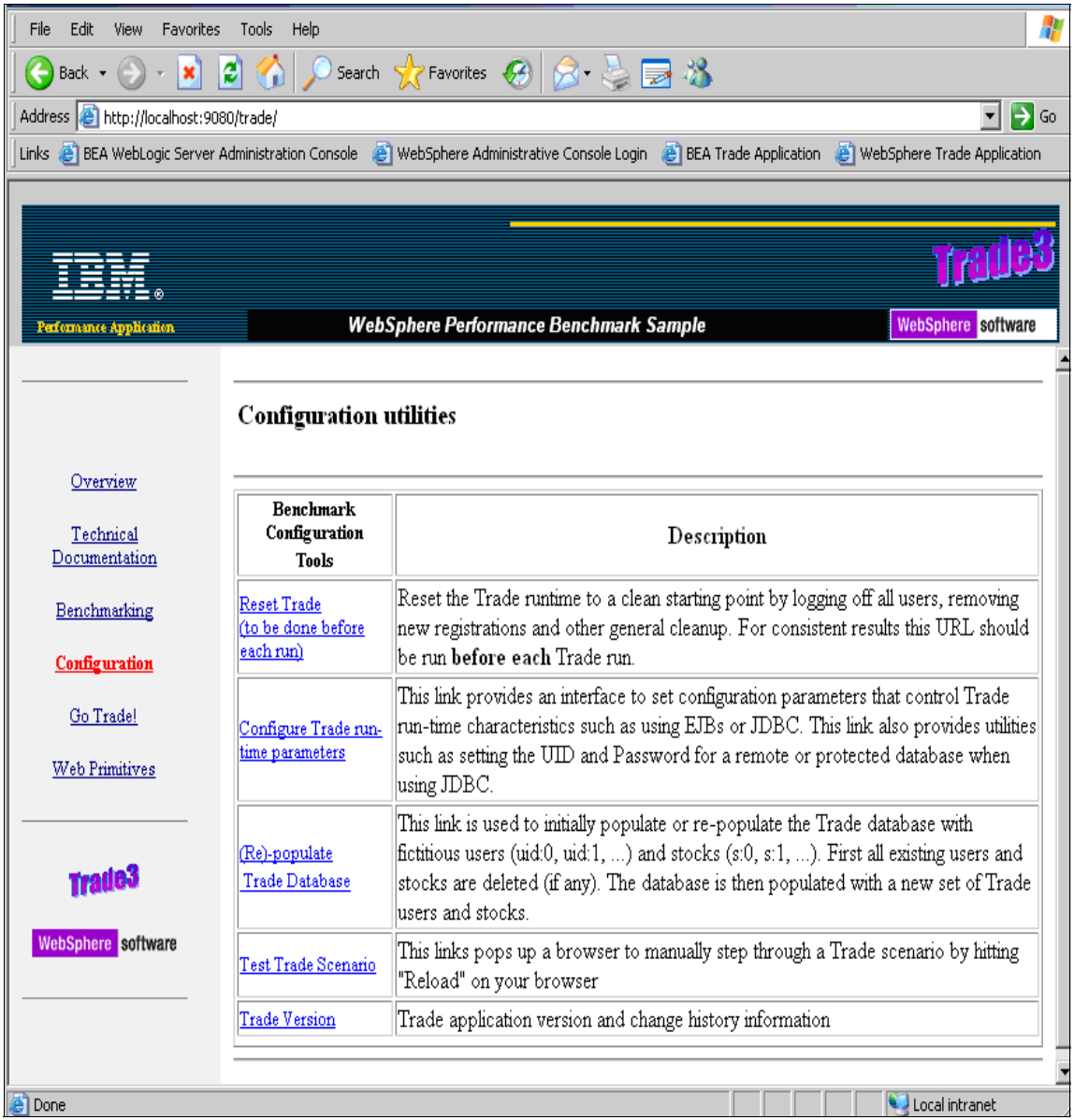


Figure 6-40 Trade Application Configuration options

2. Select the **(Re)-Populate Trade Database** option under Benchmark Configuration Tools. This populates the Trade database with fictitious users and stocks. Any existing users are deleted, and a new set of users and stocks are created. (This might take some time to complete).
3. Scroll down to the Trade Configuration section **Order-Processing Mode**. The three choices are as follows:
   - Synchronous (the default selection)
   - Asynchronous_1-Phase
   - Asynchronous_2-Phase

   The mode can be changed for subsequent runs of the Trade application by selecting another mode, and clicking **Update Config**.
4. Close this window and go back to the original Trade Application window.
5. While still in the Configuration utilities, select **Reset Trade**. Note the Order Processing mode in the table.

For the next set of tests, access the following URL:

```
http://localhost:9080/trade/scenario
```

This URL is handled by the TradeScenarioServlet. The servlet emulates a population of users by executing an action for a randomly chosen user on each access to the URL.

6.5.4 Summary

We have completed the migration of the Trade application with minimum changes to the Java code. This application was originally developed for WebSphere, then ported to WebLogic, and we ported it back to WebSphere. Trade is a fairly complex application and uses a large number of vendor-specific deployment descriptors with exactly the same code base.

We redeployed the compiled EAR file without having to change much of the Java code. We were able to demonstrate the following concepts:

- Use of the WebSphere Application Server Migration Tool V1.1 for Rational Application Developer V7.5 for automatic migration of CMP EJB mappings and vendor-specific deployment descriptors.
- Configuration of JDBC provider and data source, Service Integration Bus, JMS queues and topics and application deployment tasks with WebSphere Application Server V7.
- Basic development activities using Rational Application Developer V7.5.

6.6 xPetstore EJB migration

xPetstore EJB is an open source project hosted by SourceForge. It is a complete rewrite of Sun's PetStore J2EE reference application. xPetstore EJB was built to demonstrate J2EE development best practices and how to make use of the most popular open source frameworks such as XDoclet and Struts. We decided to include the xPetstore EJB because it is a good example of how to write portable application that run on multiple application servers and it covers many interesting and widely used technologies and frameworks:

- Message Driven beans 2.0
- Stateless Session beans 2.0
- Stateful Session beans 2.0
- Container Managed Persistence EJB 2.0
- JSP 1.2
- Servlet 2.3
- Ant build script
- XDoclet
- JUnitEE
- Struts

xPetstore also differs from most WebSphere Application Server V7 applications in the sense that it uses XDoclet extensively to generate the following artifacts:

- EJB home and business interfaces and deployment descriptors
- Application server specific deployment descriptors
- Value objects
- Deployment descriptors for servlets, servlet filters and JSP taglibs

6.6.1 Migration approach

For the xPetstore EJB application, we decided to use Rational Application Developer V7.5 and the IBM WebSphere Application Server Migration Toolkit V1.1. In this migration example, we migrate the EJB version of xPetstore from Oracle WebLogic Server 9.2 to WebSphere Application Server V7.

We have to create all the WebSphere Application Server V7-specific deployment descriptors because WebSphere Application Server V7 does not fully support xDoclet and vice versa. Creating all these deployment descriptors manually is a difficult task but by using Rational Application Developer V7.5 and the migration tool, it is relatively simple.

The migration of the xPetstore EJB application is to be done in the following sequence:

1. Build the application from source code using Ant.
2. Configure Oracle WebLogic Server 9.2 to create all necessary resources for the application, such as Data Sources, Mail provider, and so forth. Deploy xPetstore EJB into Oracle WebLogic Server 9.2 to verify that it works properly by running JUnitEE tests provided with the application.
3. Import the EAR file from WebLogic into Rational Application Developer V7.5 to take advantage of its advanced deployment descriptor editing functionality. We do not change source code, but we update resource references and export the EAR file.
4. Configure WebSphere Application Server V7 to create the same resources we created in Oracle WebLogic Server 9.2 for xPetstore EJB application.
5. Deploy the application into WebSphere Application Server V7 and rerun our JUnitEE test cases to make sure they are executed successfully.

Note: In the migration process, we remove the dependency on Ant and XDoclet. We do not modify the Ant build script to work with WebSphere Application Server V7. Rational Application Developer V7 includes all the necessary tools for making the builds. This is only done to highlight how Rational Application Developer V7 differs from using Ant and XDoclet.

It is recommended that you have an automated build process. We recommend that the build script be created from scratch after the migration is completed successfully.

6.6.2 Configuring the initial environment

The initial environment consists of xPetstore running on Oracle WebLogic Server 9.2 and using DB2 Universal Database V9.5 as its data store. xPetstore EJB officially only supports Oracle WebLogic Server 7.x but will run on Oracle WebLogic Server 9.2 without modifications. By default, xPetstore EJB does not support DB2 Universal Database V9.5, so we have included the steps required for configuring xPetstore EJB to use DB2 Universal Database V9.5.

The following list details the subsections contained in this section:

- "Getting the xPetstore EJB application" on page 168
- "Creating the sample application database" on page 168
- "Adding the DB2 JDBC driver to the class path" on page 169
- "Creating a new data source" on page 169
- "Creating a new JMS server" on page 170
- "Creating the JMS module" on page 170
- "Creating the JMS queues" on page 171
- "Creating a JMS connection factory" on page 171
- "Creating a new JavaMail session" on page 172
- "Configuring the application to use DB2" on page 173
- "Building the sample application" on page 173
- "Deploying the sample application" on page 174
- "Testing the application" on page 175

Getting the xPetstore EJB application

Before we start our migration, we need to obtain source files for the xPetstore EJB application. Download the `xpetstore-3.1.3.zip` file with all source files from the following web page by clicking the download link in the left navigation pane under Installation heading:

`http://xpetstore.sourceforge.net`

In our scenario, we downloaded and extracted the application to the following directory:

`D:\sampleApp\xpetstore-cmp-weblogic`

This directory is later referred to as <source_home>.

Creating the sample application database

In this section, we show how to create the database in DB2 Universal Database V9.5 and verify the connection.

1. From a command line, start DB2 CLP (command line processor) by entering the **db2cmd** command.
2. Execute the following commands to create the xPetstore EJB database and to verify connectivity:

   ```
   db2 create db wl_xp_ej
   db2 connect to wl_xp_ej user db2admin using db2adm1n
   db2 disconnect all
   ```

   At this point, the database is empty, we still need to create the necessary tables and load the sample data into the tables. We do this last before we deploy the application.

Next, we configure the resources necessary to run the xPetstore EJB application on Oracle WebLogic Server 9.2.

Adding the DB2 JDBC driver to the class path

For this migration, we created a new WebLogic domain called *xpetstore-cmp* to have a clean starting environment. Therefore, we had to repeat the procedures described in the migration of the Trade application. The details of how to add the DB2 drivers are covered in “Adding the DB2 JDBC driver to the WebLogic class path” on page 108.

Creating a new data source

Perform the following to create a new data source:

1. In the WebLogic server console, navigate to **xPetstore** → **Services** → **JDBC** → **Data Sources**. Create a new data source with the values listed in Table 6-26.

Table 6-26 JDBC Data Source configuration

Parameter	Values
Database type	DB2
Database driver	IBM DB2 Driver (Type 2 XA) Versions:7.X,8.X
Name	Petstore JDBC Datasource
JNDI Name	jdbc/xpetstore

2. Click **Next**.

   The Transaction Options window is displayed and an informational message is displayed stating that based on the selected driver type, only XA 2 phase commit transactions supported.

   Click **Next**.

3. In the Connection Properties window enter the information in Table 6-27 in to the database.

Table 6-27 Database Connection Properties

Parameter	Value
Database Name	wl_xp_ej
Host Name	localhost
Port	50000
Database User Name	db2admin
Password	db2adm1n

4. On the Test Database Connection page, click **Test Configuration**. The message `Connection test succeeded` is displayed.
5. On the Select Targets page, select **AdminServer** as the target server and click **Finish**.

Creating a new JMS server

Perform the following steps to create the JMS resources.

1. From the WebLogic Server Console home page, create a new JMS Server by navigating to **xPetstore** → **Services** → **Messaging** → **JMS Servers**. Use the parameters in Table 6-28 to create the JMS server.

Table 6-28 JMS Server configuration

Parameter	Value
Name	Petstore JMS Server
Target	AdminServer

Creating the JMS module

Perform the following steps to create a JMS module to act as the container to defining JMS queues and connection factory:

1. Navigating to **xPetstore** → **Services** → **Messaging** → **JMS Modules.**
2. Enter `JMS Resource` for the JMS Module name and click **Next**.
3. Select **AdminServer** as the target and click **Finish**.

Creating the JMS queues

Next, we create the JMS queues and connection factories needed by the XPetstore application at run time.

1. In the JMS Resource Module, click **JMS Resource** and click **New**. Configure the queue destinations by creating a new JMS Queue using the values in Table 6-29. Click **Next**

Table 6-29 Order MDB's JMS queue configuration

Parameter	Value
Name	Order queue
JNDI name	jms/queue/order

2. In the Subdeployment drop-down menu, select **PetstoreJMSSubDep**. In the Targets field, make sure **Petstore JMS Server** is selected. Click **Finish**.

   This queue is used by the Order MDB in xPetstore EJB.

3. Create another JMS queue to be used by the Mailer MDB. Create a new JMS queue in the same way as described earlier, specifying the values in Table 6-30 on page 171.

Table 6-30 Mail MDB's JMS queue configuration

Parameter	Value
Name	Mail queue
JNDI name	jms/queue/mail

Creating a JMS connection factory

Use the following steps to create a JMS connection factory.

1. Click **Services** → **Messaging** → **JMS Modules** → **JMS Resource** and configure a new JMS connection factory with the values shown in Table 6-31.

Table 6-31 JMS connection factory configuration

Parameter	Value
Name	PetStore JMS Connection Factory
JNDI name	jms/ConnectionFactory

2. In the Targets field, click **Advanced Targeting**.

3. On the Subdeployment drop-down menu, select **PetstoreJMSSubDep**. In the Targets field make sure **Petstore JMS Server** is selected. Click **Finish**.
4. On the JMS connection factory page, click the Transactions tab and select **XA Connection Factory Enabled**.
5. Click **Save**.

Creating a new JavaMail session

Perform the following steps to create a new JavaMail session:

1. From the WebLogic Server Console, navigate to **Services** → **Mail sessions**.
2. Create a new JavaMail session with the values listed in Table 6-32.

Table 6-32 JavaMail session configuration

Parameter	Value
Name	PetStore Mail Session
JNDI name	mail/MailSession

JavaMail Properties	mail.from=admin@localhost;
	mail.transport.protocol=smtp;
	mail.smtp.host=localhost;
	mail.store.protocol=pop3;
	mail.pop3.host=localhost;
	mail.user=admin

Target	AdminServer

Configuring the application to use DB2

This section describes how to configure the xPetstore application to use DB2 Universal Database V9.5 because xPetstore does not support DB2 by default.

To use DB2 Universal Database V9.5, you have to make changes in the configuration files of the application. This changes are necessary because the application is using Hibernate to map the application objects with the relational data, and it needs to know which database it is accessing, and other configuration data, such as usernames and passwords and driver type.

Follow these steps to configure xPetstore Servlet to use DB2 Universal Database V9.5:

1. Edit the `<source_home>\conf\db\database.properties` by changing the database property to `database=db2`.
2. In the same directory, create a file called `db2.properties` such as the one shown in Example 6-2.

Example 6-2 xPetstore database configuration file

```
#=========================================================
# DB2 configuration
#=========================================================
db.driver=COM.ibm.db2.jdbc.DB2XADataSource
db.url=jdbc:db2:wl_xp_ej
db.user=db2admin
db.password=db2adm1n
db.classpath=${lib.dir}/db2java.zip:${lib.dir}db2jcc.jar:${lib.dir}db2j
cc_license_cu.jar
db.foreign.key=true

hibernate.dialect=cirrus.hibernate.sql.DB2Dialect
hibernate.generator.class=native
hibernate.outer.join=true
```

Building the sample application

This section describes the necessary steps for making a build of xPetstore EJB that runs on the Oracle WebLogic Server 9.2 platform. xPetstore EJB uses Ant build scripts for making builds. The build script consists of the following files:

- `<source_home>\xpetstore-ejb\build.xml`
- `<source_home>\xpetstore-ejb\build-weblogic.xml`

The build script is configured by modifying the following property files:

- `<source_home>\conf\as\appserver.properties`
- `<source_home>\conf\as\weblogic.properties`

To make the build, follow these steps:

1. Change the xPetstore EJB target platform to WebLogic by changing the app.server property to `weblogic` in the `<source_home>\conf\as\appserver.properties` file.
2. Build xPetstore EJB by running `build.bat` from the `<source_home>\xpetstore-ejb` directory. Check the value of your JAVA_HOME variable and reset it in the build.bat file if necessary.

Ant uses the default target **all** to build the application. This makes Ant perform the following tasks:

1. Clean the build directory by removing any temporary files.
2. Run XDoclet to generate source code and deployment descriptors.
3. Compile all source code.
4. Create a jar file containing EJBs.
5. Create a WAR file containing the web application.
6. Create an EAR file `<source_home>\dist\xpetstore-ejb.ear` containing the EJB and WAR file.

The EAR file contains the following files that make up the complete xPetstore EJB application:

- `xpetstore-ejb.jar`
- `xpetstore-ejb.war`
- `xpetstore-ejb-test.war`

Deploying the sample application

This section describes how to deploy the EAR package containing the xPetstore EJB application to Oracle WebLogic Server 9.2.

1. Copy the `xpetstore-ejb.ear` file from `<source_home>\dist\` folder to the `<weblogic_home>\user_projects\domains\xpetstore\autodeploy` directory:

> **Note:** Oracle WebLogic Server 9.2 automatically deploys the application if it started in development mode.
>
> Oracle WebLogic Server 9.2 also creates the database schema needed by the CMPs if the WebLogic instance is running in development mode. In this case, WebLogic adds an extra column called *wls_temp* to each container-created tables.

2. Populate the database with the sample xPetstore EJB data by opening a command line window and executing the following commands.

```
db2cmd
db2 connect to wl_xp_ej user db2admin using db2adm1n
db2 -tvf <source_home>xpetstore-ejb\sql\xpetstore-cmp-wls.data.sql
```

Important: Because our WebLogic domain was created to run in development mode, when we deploy the application it creates the tables with an extra column. Hence, the `data.sql` provided by the xPetstore EJB application does not work.

See Appendix C, "Additional material" on page 375 for instructions on how to download `xpetstore-cmp-wls.data.zip`. Extract the file into the `<source_home>xpetstore-ejb\sql` directory.

The difference from the original `data.sql` file is that this one contains the column names where the data is inserted. This is needed because WebLogic changed the container-created tables.

3. Access the application at the following URL:

```
http://localhost:7001/xpetstore-ejb
```

At this point, you have the database with all the sample data populated. The next step is to test the application.

Testing the application

In this section, we show you how to test the xPetstore EJB application deployed on Oracle WebLogic Server 9.2. Testing is straightforward, because the EJB version of xPetstore includes test cases implemented with JUnitEE.

1. To run the test cases go to the following URL:

   ```
   http://localhost:7001/xpetstore-test/index.html
   ```

   You should see a JUnitEE test page displayed in your browser.

2. Select the tests options you want to test and click **Run**. On the next page, you should see no errors.

6.6.3 Migrating the sample application

In this section, we describe the steps we performed to migrate xPetstore EJB. This migration effort is straightforward because xPetstore EJB was developed following best-practices and J2EE standards.

Importing the application EAR file

Start the migration by importing the EAR file that was built for and deployed to Oracle WebLogic Server 9.2 into Rational Application Developer V7.5.

Note: We used a new Rational Application Developer V7 workspace for the migration. The workspace folder is referred to as <rad_workspace>.

To import the EAR file start Rational Application Developer V7.5 and perform the following steps:

1. In Rational Application Developer Java EE Enterprise Explorer perspective, right-click, and from the context menu, select **Import** → **EAR file**. See Figure 6-41 on page 176.

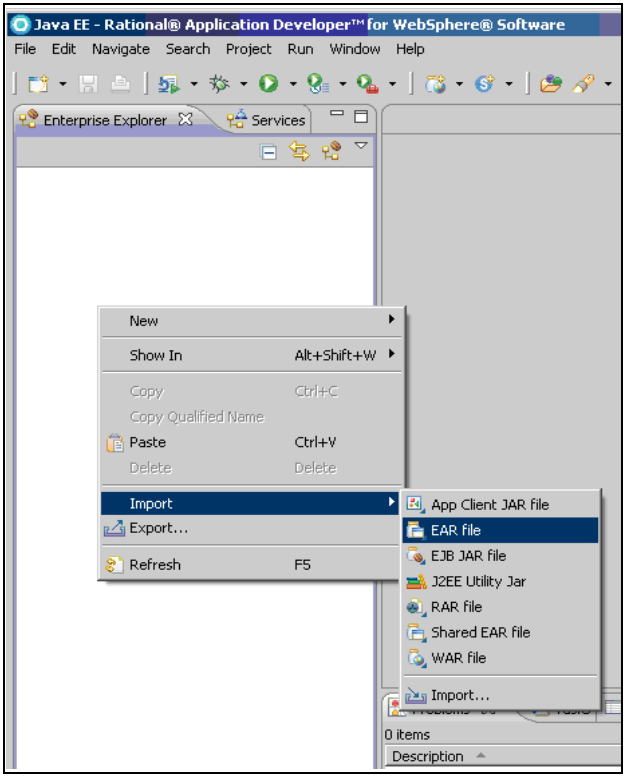


Figure 6-41 Import EAR file

2. Click **Browse** and select the `xpetstore-ejb.ear` EAR file. In our example we have it is located in `<source_home>\xpetstore-ejb\xpetstore-ejb.ear`.
3. Make sure the Target runtime is **WebSphere Application Server v7**. Click **Next**.
4. In the next window do not select any of utility jars and web libraries. Those are included in the EAR upon import.
5. In the Ear Module and Utility JAR Projects, make sure all modules are selected and click **Finish**.

The wizard automatically created four new projects in your workspace as shown in Figure 6-42 on page 177.

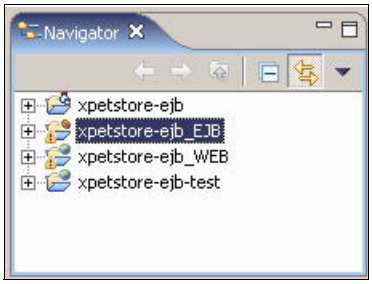


Figure 6-42 Importing xPetstore Application

The first item in the navigator view is the EAR project. The second item contains the EJB project and the third item the web application project. The last item is the test project, which contains unit tests.

Tip: Before proceeding, configure Rational Application Developer V7 not to build the projects automatically after changes to the source. This is done by clearing the **Build Automatically** option in the Project menu.

Importing the application source code

xPetstore EJB was built following best practices and uses separate packages for the EJB and web application projects, which makes it easier to import the source code.

xPetstore uses XDoclet extensively to generate source code and deployment descriptors. The deployment descriptors were included in the EAR file.

Import the Java source files that were not generated by XDoclet by performing the following steps:

1. Copy the EJB project's source code folder, located in the `<source_home>\xpetstore-ejb\java\xpetstore` directory, to the xpetstore-ejb_EJB project's `ejbModule\xpetstore` folder in the Rational Application Developer V7.5 workspace. You can use the command line, copy and paste, drag and drop, or any other tool to copy files.
2. Copy the WAR project's source code folder, located in the `<source_home>\xpetstore-ejb\web\xpetstore`, directory to the xpetstore-ejb_WEB project's `src\xpetstore` folder in the Rational Application Developer V7.5 workspace.
3. Copy the test project's source code folder, located in the `<source_home>\xpetstore-ejb\test\xpetstore` directory, to the xpetstore-ejb-test project's `JavaSource\xpetstore` folder in the Rational Application Developer V7.5 workspace.
4. In the Navigator view, select all four projects, right-click and select **Refresh**. This refreshes the project structure from the file system and displays the files you just copied.
5. Click **Project** → **Build All** in the menu. Rational Application Developer V7.5 reports errors in the **Problems** view for the WAR and test projects because it cannot find the source code for the EJBs.
6. Fix the errors by specifying that the project depends on the EJB project. Right-click the xpetstore-ejb_WEB project and choose **Properties** from the menu.
7. From the Properties window, select **Java Build Path** → **Projects** and select the xpetstore-ejb_EJB project from the list. Click **OK**.
8. The test project also references code located in the EJB project, so repeat steps 5 through 7 for the test project.
9. Click **Project** → **Build All** from the menu. The Problems view in Rational Application Developer V7.5 now contains only warnings about imports that are not used.

   You should still see two broken links warnings; we do not fix these because they are trivial.

XDoclet automatically generates the source code for EJB local and remote interfaces and the value objects when we create an xPetstore EJB build. This source code must be imported separately because it is located in a separate folder. To do so, perform the following steps:

1. Open the `ejbModule` folder in the EJB project. For each class file in the folder, verify that there is a corresponding Java source file in the same directory. If the source file is missing, it means that the files generated by Ant using XDoclet needs to be copied from the `<source_dir>\xpetstore-ejb\build\java` directory to the xpetstore-ejb_EJB EJB project's corresponding directories.
2. Once finished, click **Project** → **Build All** in the menu. You should see no errors in the Problems view.

The xPetstore EJB source code is now imported into the new development environment. You can now remove all XDoclet tags from the source code and create a new Ant script to automate the build script.

Creating a new back end

When you first import the EAR file into Rational Application Developer V7.5, the project is configured to use Cloudscape as the back end database. Our initial and destination environments use DB2 Universal Database V9.5. In the source environment, the database schema was created for Oracle WebLogic Server 9.2. We shall use the same database but create a new schema designed for WebSphere Application Server V7.

There are three options available in Rational Application Developer V7.5 for mapping CMPs to a database.

- Bottom-up

  This option generates EJBs and the mapping from an existing database.

- Top-down

  Generates a database schema and map from existing EJBs.

- Meet-in-the-middle

  Generates the mapping between an existing database and existing EJBs.

Note: We used the same database as Oracle WebLogic Server 9.2 but will create a new schema that is used by WebSphere Application Server V7.

We did not use the database schema generated for Oracle WebLogic Server 9.2. This is because the table and column names do not match the CMPs. In a real-world migration project, the existing database is most likely used without modifications. For that scenario, use the meet-in-the-middle approach.

Generate a new database schema for xPetstore EJB by using the top-down approach, using the following procedure:

1. In the Enterprise Explorer view in Rational Application Developer V7.5, right-click the **xpetstore-ejb_EJB** project.
2. Navigate to **Java EE** → **EJB to RDB Mapping(1.x-2.x)** → **Generate Map** from the context menu.
3. Choose **Create a new backend folder** and click **Next**.
4. Select **Top-Down** from the list of options and click **Next**.
5. On the next page, specify the values in Table 6-33. Click **Finish**.

Table 6-33 Database configuration for wizard

Parameter	Value
Target database	DB2 for Linux, UNIX and Windows V9.5
Database name	wl_xp_ej
Schema name	NULLID

You can see the same folder structure in the Navigator view, as in Figure 6-43.

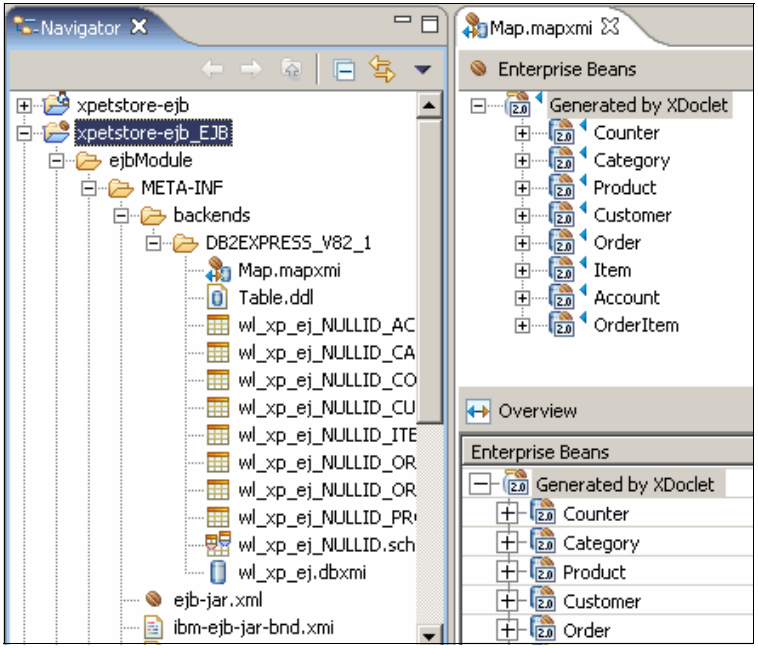


Figure 6-43 Database mapping generated with Top-down option

6. Verify that the back end is active by double-clicking the `ejb-jar.xml` file in the Navigator view. This opens the Deployment Descriptor Editor.
7. On the Overview tab, scroll down to the **Backend ID** section and verify that it is using DB2 V9.5.

In the following section, we use the `Table.ddl` file that was generated by the wizard to create the database schema.

Creating a new database schema

The next step is to create the database by importing the database schema that was created using the EJB to RDB Mapping wizard described previously.

Create the database schema in the wl_xp_ej database by following these steps:

1. Open a DB2 command line processor (CLP) by executing the **db2cmd** command from a command line window.
2. From the DB2 CLP, connect to the DB2 database and create the database schema with the following commands:

```
db2 connect to wl_xp_ej user db2admin using db2adm1n
db2 -tvf
<rad_workspace>\xpetstore-ejb_EJB\ejbModule\META-INF\backends\DB2UDB
NT_V95\Table.ddl
```

Note: DB2 Universal Database V9.5 reports the error shown in Example 6-3. The error is caused by the CUSTOMER table, which is too big to fit in the default table space. To fix this, create a new table space with a page size of at least 8 KB. This problem can also be fixed by making the columns smaller. For example, open the file `Table.ddl` file. Do a search for `250` and replace with `150`.

Example 6-3 Database error

```
DB21034E The command was processed as an SQL statement because it was
not a
valid Command Line Processor command. During SQL processing it
returned:
SQL0286N A default table space could not be found with a page size of
at
least "8192" that authorization ID "DB2ADMIN" is authorized to use.
SQLSTATE=42727
```

You need to import the xPetstore data. We used a modified version of the original `data.sql` file because the database column names have changed in this new schema.

1. Open the `<source_home>\xpetstore-ejb\sql\data.sql` file in an editor.
2. Rename all the table names to match the new schema (`Table.ddl`). This means removing `T_` from all table names.
3. Change the LISTPRICE and UNITCOST columns' datatype, because they are defined as DOUBLE in the new database schema and as VARCHAR in the data script.

   This is done by removing the single quotes from all the **`INSERT`** commands for the ITEM table, so that the column values are defined as illustrated in Example 6-4.

Example 6-4 Updated INSERT statement

```
INSERT INTO ITEM VALUES ('EST-1', 'Large', 16.50, 10.00, 'fish1.jpg',
'FI-SW-01');
```

4. Rename the modified file to `data2.sql`
5. Import the data by issuing the following commands from a DB2 CLP window:

```
db2 connect to wl_xp_ej user db2admin using db2adm1n
db2 -tvf <source_home>\xpetstore-ejb\sql\data2.sql
```

Running the Application Migration Tool

We have the source code imported into our project and the database schema created, so the next step is to run the Application Migration Tool to analyze the source code, jsp's, and deployment descriptors. Perform the following steps to analyze the source code.

1. Configure the migration tool by following the steps outlined in "Migrating the sample application" on page 113.
2. Click **Analyze** to analyze all the project in the workspace.

   The Software analyzer view is opened at the bottom of the window. It contains four tabs. Each tab corresponds to one of the analysis domains selected based on the Rule Set selection you made earlier.

   – The Class-Path Review tab list any class path related issues.
   – The Java Code Review tab lists Java code related issues.
   – The XML File Review tab lists issues with deployment descriptors and the presence of any web logic specific files.
   – The JSP Code Review tab lists issues found in JSP files, commonly associated with web projects.

3. Click the JSP File Review tab. See Figure 6-44.

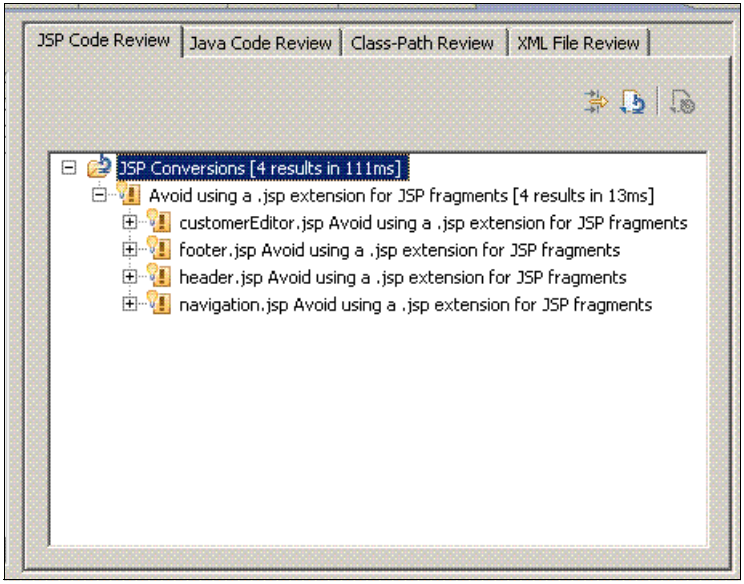


Figure 6-44 Migration toolkit JSP Code Review

4. Expand the JSP Conversion tree and right-click the **customerEditor.jsp Avoid using a .jsp extension for JSP fragments** rule violation. From the context menu select **QuickFix**. The **Refactoring** dialog box (Figure 6-45) is displayed.

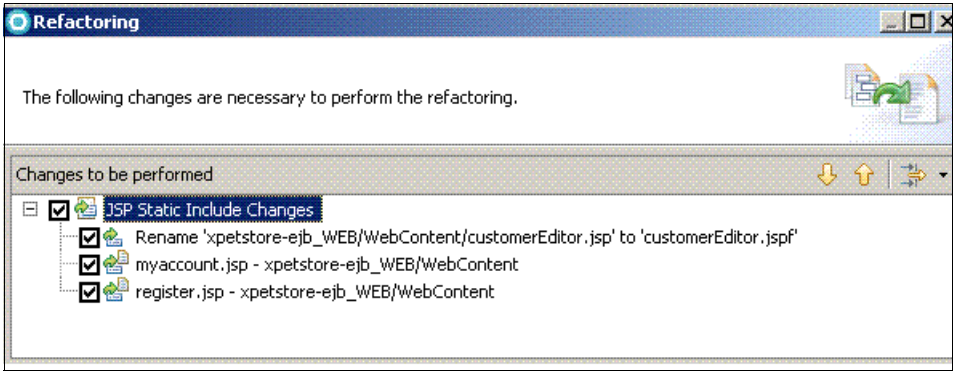


Figure 6-45 Refactoring JSP file fragments

5. Select the first fix to rename the file and click **Finish.**
6. Repeat step 5 for the remaining three JSP files.

7. Click the XML File Review tab and expand the **Use WebSphere bindings to define EJB JNDI names** category. See Figure 6-46.

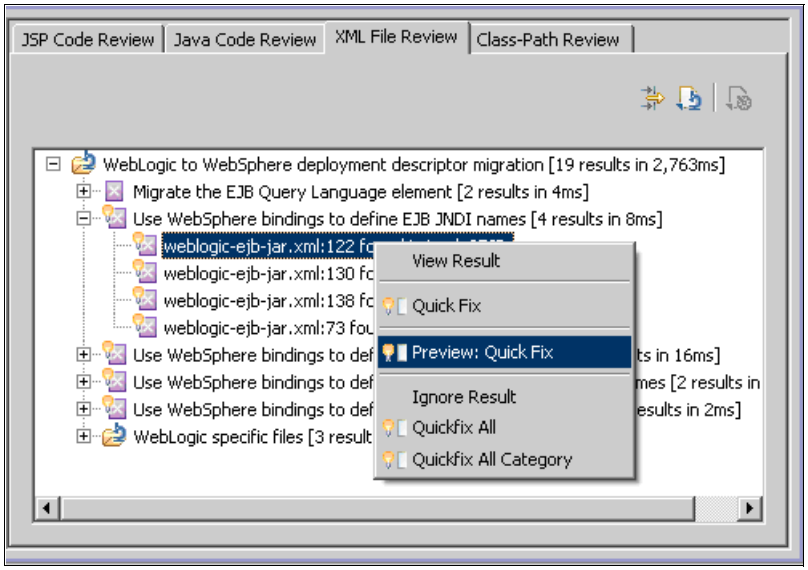


Figure 6-46 XML File Review tab

8. Right-click the `weblogic-ejb-jar.xml` entry file. From the context menu, select **Quick Fix.** This creates the appropriate JNDI names in the corresponding WebSphere binding file.
9. Repeat step 8 for the remaining rules under the same category.
10. Still within the XML File Review tab, click the **Use WebSphere binding to define EJB reference names** category. From the context menu select **Quickfix All**.
11. Repeat step 10 for the **Use WebSphere bindings to define Message Driven beans JNDI names** and **Use WebSphere bindings to define resource reference names** categories.
12. Right-click the **Use WebSphere extensions to define CMP mappings** rule and select **Ignore Results.** We are ignoring this rule because we already created the back-end folder and mapped the CMP beans to their corresponding database tables and columns.
13. Handle the WebLogic specific EJB query tags. The migration tool does not provide a quick fix for converting WebLogic EJB query tags to the corresponding WebSphere at this point.

14. Review the help for the WebLogic Query language constructs by highlighting one of the rules and pressing the F1 function key. This launches the help. Click **See Details** link from the help.

15. Double-click the first rule under the **Do not use WebLogic specific EJB Query language** category. This shows you the weblogic-ql construct. You should see the token AS that is used. It is a WebLogic construct and not an ejb-ql construct. The token needs to be removed from the query. See Figure 6-47.

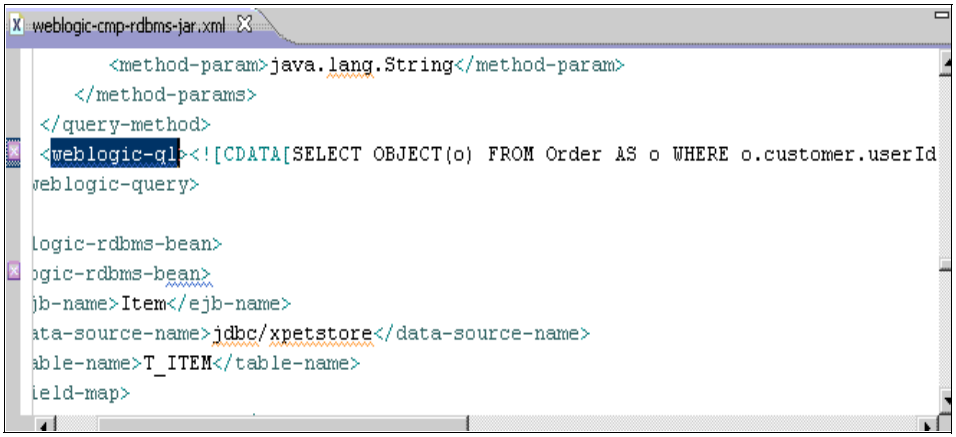


Figure 6-47 WebLogic specific Query Language tag

16. Remove the “AS” from the query and save the file.

17. Repeat this step for the remaining two occurrences in the file as indicated by the migration tool.

18. Return to the analysis tab and right-click the **weblogic-ql** rules and select **Ignore Result**. This removes it from the view. It also enters a comment in the `weblogic-cmp-rdbms-jar.xml` file to tell the analyzer to ignore the next iteration.

19. Save and close the deployment descriptor file.

20. n the XML Review tab, expand the **WebLogic specific files** category and expand all sub-options. Those are the WebLogic-specific deployment descriptors files. Delete them, because we migrated all of their entries to the corresponding WebSphere binding files.

21. n the Enterprise Explorer view, expand the EJB project and browse to `/xpetstore-ejb_EJB/ejbModule/META-INF` folder and delete the following files:

    - `weblogic-ejb-jar.xml`
    - `weblogic-cmp-rdbms-jar.xml`

Migrating xpetstore J2EE Specification Level

Because this application is J2EE 1.3, we shall migrate this application's deployment descriptors to J2EE 1.4. Review each window before proceeding to the next step.

1. In the Java EE perspective right-click **xpetstore-ejb**. From the context menu, select **Java EE** → **Specification Migration Wizard**.

   The next window provides informational messages before running the J2EE specification migration wizard. Click **Next.**

2. In the J2EE version drop-down menu, select **1.4** and click **Next**.
3. In the EJB Module Migration window, make sure **xpetstore-ejb_EJB** is selected and click **Next**.
4. In the Web Projects window make sure that both **xpetstore-ejb_WEB** and **xpetstore-ejb-test** are selected and click **Finish**.

   The Migration Complete dialog box is displayed.

Note: Click **Details** on the Migration Complete dialog box to see the exact changes the migration wizard performed against the project.

5. Click **OK**.

Configuring the mail provider in WebSphere

The JavaMail session provides an abstraction layer for communicating with an email server. This section describes how to configure a mail provider, which is used to create a JavaMail session in WebSphere.

1. Start the WebSphere Application Server V7 by entering the following commands:

   ```
   cd <was_home>\runtimes\base_v7\profiles\<profile_name>\bin
   startserver server1
   ```

2. Open the WebSphere Administrative Console by accessing the following URL:

   ```
   http://localhost:9060/ibm/console
   ```

3. Select **Resources** → **Mail** → **Mail Providers** from the left navigation pane.
4. In the right pane, select the **Server scope** in the scope drop-down menu.

5. Click the **Built-in Mail Provider** link. Click **Mail Sessions** in the Additional Properties section. Create a new mail session using the values in Table 6-34.

Table 6-34 JavaMail session configuration

Parameter	Value
Name	xPetstoreMail
JNDI name	mail/xpetstore/MailSession
Server	localhost
Protocol	SMTP

6. Click **OK** to create the mail session.

Creating a new data source

xPetstore uses a JDBC data source for connecting to the database. To set up the data source, we need to configure a JDBC provider, a J2C alias for storing the database authentication information and the data source itself.

1. From the administration console, select **Environment** → **WebSphere Variables.**
2. In the right frame select **Node=<node_name>** in the scope drop-down menu.
3. Click the **DB2UNIVERSAL_JDBC_DRIVER_PATH** environment variable.
4. Specify the value, which is the correct path to the DB2 JDBC driver and click **OK**. In our migration example, the path to the JDBC driver is `C:\IBM\SQLLIB\java.`
5. Click **OK**.
6. Save the changes.
7. Click the **UNIVERSAL_JDBC_DRIVER_PATH** environment variable and specify the value **${WAS_INSTALL_ROOT}/universalDriver/lib.** This points to the `db2jcc_license_cu.jar` which is the DB2 license file.
8. Click **OK** and save the changes.

Create the JDBC provider, data source and J2C authentication alias. The steps to do this follow.

1. From the administration console, select **Resources → JDBC → DBC providers.**
2. In the Scope drop-down menu select **Node=<Node-name**>.
3. Click **New** to create a new JDBC provider using the values in Table 6-35.

Table 6-35 JDBC provider configuration

Parameter	Value
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	XA datasource

4. Save the changes and double click the JDBC provider you just created.
5. Click **Data sources** in the Additional properties section and create a new data source with the characteristics shown in Table 6-36.

Table 6-36 JDBC data source configuration

Parameter	Value
Name	Petstore Datasource
JNDI Name	jdbc/xpetstore
Driver type	4
Host name	localhost
Port	50000
Database name	wl_xp_ej

6. Click **Finish**. Save the changes.
7. Click **JAAS – J2C authentication data** in the Related Items section.

8. Click **New**, and create a new alias using the information shown in Table 6-37.

Table 6-37 J2C Alias Configuration

Parameter	Value
Alias	xpetstoreAuthData
User ID	db2admin
Password	db2adm1n

9. Click **Apply**.
10. Click **OK** and go back to the data source you just created. From the **Component-managed authentication alias** drop-down menu, select the J2C alias you created.
11. Click **Apply**. Save your changes.
12. Verify the connection to the database by selecting the data source you just created from the list and clicking **Test connection**.

Configuring JMS

WebSphere Application Server V7 comes has default messaging provider that is based on JMS specification 1.1. We use the default messaging provider to create the messaging resources required by the xpetstore application.

This section has several steps, broken in to the following subsections:

- "Creating a new service integration bus" on page 190
- "Creating a bus member" on page 190
- "Creating a bus destination" on page 190
- "Creating a JMS connection factory" on page 191
- "Creating JMS queues" on page 191
- "Creating a JMS activation specification for the JMS queues" on page 192
- "Configuring Message Driven beans" on page 193

Creating a new service integration bus

The first step in configuring JMS messaging in WebSphere Application Server V7 is to create an service integration bus. Perform the following steps:

1. Open the administration console. Select **Service Integration** → **Buses** in the navigation pane on the left.
2. On the following page, create a new ESB. Specify `bus2` as the name. Click **OK**.
3. In the Summary window, click **Finish.**

Creating a bus member

The bus has to be deployed to a bus member. The members of a service integration bus are the application servers in which messaging engines for that bus can run. Perform the following steps:

1. From the administration console, select **Service Integration** → **Buses** in the navigation pane on the left. Select the bus we created in “Creating a new service integration bus” on page 190.
2. Click **Bus members** in the Topology section.
3. Click **Add** and select the server to which the bus is deployed. Accept all defaults on the remaining windows and click **Finish**.
4. Save the changes you made to the configuration.

Creating a bus destination

The destination is the target of all messages sent to the Information Service Bus. To create the bus destination, follow these steps:

1. Select the bus created in “Creating a new service integration bus” on page 190.
2. Click **Destinations** in the Destination resources section. Create a new Queue destination. Specify **xPetstoreQueue** as the identifier and deploy it to the bus member created.
3. Click **Finish** and save your configuration.

You have now configured the messaging bus, bus member and bus destinations, which form the infrastructure for JMS messaging. The next step is to configure the JMS connection factory and queues.

Creating a JMS connection factory

Before configuring the JMS queues, configure a JMS connection factory, which is used by the xPetstore EJB MDBs to create a connection to the JMS provider.

1. From the administration console, select **Resources** → **JMS** → **JMS providers** from the left navigation pane.
2. In the right frame, select **Node=<node_name>,Server=server1** in the scope drop-down menu.
3. Click **Default messaging provider**.
4. Click **Queue connection factories** in the Additional Properties section.
5. Create a new JMS queue connection factory with the characteristics shown in Table 6-38.

Table 6-38 JMS queue connection factory configuration

Parameter	Values
Name	xPetstore JMS connection factory
JNDI name	jms/xpetstore/QueueConnectionFactory
Bus name	bus

6. Click **OK** and save the changes you made.

Creating JMS queues

xPetstore uses one Message Driven bean for order processing and one for sending emails. The MDBs require that we set up two JMS queues. Perform the following steps to do so:

1. In the right frame, click **Default messaging provider**.
2. Click **Queues** in the Additional Properties section.
3. Click **New**, and create a new JMS queue using the information in Table 6-39.

Table 6-39 Mail queue configuration

Parameter	Value
Name	xPetstore Mail Queue
JNDI name	jms/queue/mail
Bus name	bus
Queue name	xPetstoreQueue

4. Click **OK** and create the second JMS queue in the same way with the characteristics shown in Table 6-40.

Table 6-40 Order queue configuration

Parameter	Value
Name	xPetstore Order Queue
JNDI name	jms/queue/order
Bus name	bus
Queue name	xPetstoreQueue

Creating a JMS activation specification for the JMS queues

The last thing to configure in WebSphere Application Server V7, related to JMS, is the JMS activation specification that binds the MDBs to the default JMS messaging provider.

The JNDI name for the JMS activation specification that you create is used in the MDB's deployment descriptor to bind the MDB to the messaging queue.

Perform the following steps to create a JMS activation specification for the JMS queues:

1. From the WebSphere Administrative Console, select **Resources** → **JMS. Providers** → **Default messaging** from the navigation pane on the left.
2. Click **Activation Specification** in the Additional Properties section.
3. Create a new JMS activation specification with the values in Table 6-41.

Table 6-41 Order MDB's JMS activation specification configuration

Parameter	Values
Name	xPetstore Order Activation Specification
JNDI name	eis/xPetstoreOrderActivation
Destination JNDI name	jms/queue/order
Bus name	bus

4. Click **OK** and save your changes.

5. Create a second JMS activation specification in the same way with the characteristics shown in Table 6-42.

Table 6-42 Mailer MDB's JMS activation specification configuration

Parameter	Value
Name	xPetstore Mail Activation
JNDI name	eis/xPetstoreMailActivation
Destination JNDI name	jms/queue/mail
Bus	bus

You now have all the necessary components for JMS messaging configured. The next step is to configure the MDBs to use the JMS activation specification and queues.

Configuring Message Driven beans

In the previous section, we described how to configure all the server-side components related to JMS. In this section, we describe how to use the Deployment Descriptor Editor in Rational Application Developer V7.5 to configure the MDBs to use those components. This is done by specifying the correct JMS Activation Specification and JMS queues for each MDB in the project.

The Deployment Descriptor Editor is used to edit the standard `ejb-jar.xml` deployment descriptor and to create the WebSphere Application Server V7 specific deployment descriptors. The `ejb-jar.xml` and the Oracle WebLogic Server 9.2 specific deployment descriptors were originally generated by XDoclet.

Perform the following steps to configure the message driven beans

1. Open Rational Application Developer V7.5 and browse to the EJB project xpetstore-ejb_EJB.
2. Open the Deployment Descriptor Editor by double-clicking the `ejb-jar.xml` file located in the `/xpetstore-ejb_EJB/ejbModule/META-INF` folder in the Enterprise Explorer view.

3. Select the bean tab. You should see a window similar to Figure 6-48.

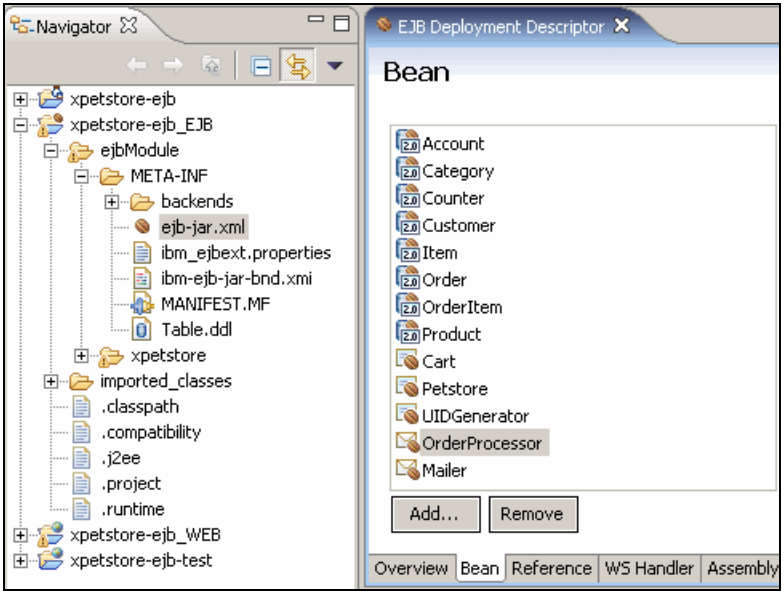


Figure 6-48 EJB Deployment Descriptor Editor

4. Select the **OrderProcessor** MDB from the list of EJBs. In the WebSphere Bindings section, select **JCA Adapter** and specify the values shown in Table 6-43.

Table 6-43 OrderProcessor MDB's WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/xPetstoreOrderActivation
Destination JNDI name	jms/queue/order

You have now configured the MDB to use the JCA activation specification and Destination you configured in the previous section.

Tip: Every time you make changes in the Deployment Descriptor, save those updates by pressing Ctrl+S.

5. Select the **Mailer** MDB in the list of EJBs. In the WebSphere Bindings section, select **JCA Adapter** and specify the values in Table 6-44.

Table 6-44 Mailer MDB's WebSphere binding configuration

Parameter	Value
ActivationSpec JNDI name	eis/xPetstoreMailActivation
Destination JNDI name	jms/queue/mail

You have now completed the configuration of the MDBs and JMS messaging.

Configuring Session beans and Container Managed beans

This section describes how to configure the Session beans and Container Managed beans to run on WebSphere Application Server V7. The only changes to be made are the application specific deployment descriptors that were generated by XDoclet for Oracle WebLogic Server 9.2. In our migration example, we used the Deployment Descriptor Editor in Rational Application Developer V7.5 to generate the WebSphere Application Server V7 specific deployment descriptors.

Specifying EJB JNDI names

The first step is to assign JNDI names to the EJBs. This needs to be done because the EJB lookups are done through resource references specified in web.xml and not the JNDI name directly.

1. Open the Deployment Descriptor Editor by double-clicking the `ejb-jar.xml` in the Enterprise Explorer view.
2. Click the bean tab.
3. Select **AccountEJB** and specify the values shown in Table 6-45 in the WebSphere Bindings section.

Table 6-45 WebSphere bindings for the Account EJB

Parameter	Value
JNDI name	ejb/AccountLocal

4. Repeat the previous step for each EJB in the list, except for the MDBs. Specify a unique JNDI name for each EJB in the following format:

   `ejb/<BeanName>Local`

Configuring the default data source for CMPs

Specify a default data source that is used by all CMP beans.

1. Select the Overview tab in the Deployment Descriptor Editor.
2. In the WebSphere Bindings section, under the JNDI - CMP Connection Factory Binding section, specify the values shown in Table 6-46.

Table 6-46 Default data source configuration

Parameter	Value
JNDI name	jdbc/xpetstore

Configuring EJB resource references

xPetstore EJB requires a JDBC data source, JMS queues, and factories that are all managed by WebSphere Application Server V7. The links to these resources are declared in the `ejb-jar.xml` and WebSphere Application Server V7 specific deployment descriptors using resource references.

1. Open the Reference tab in the Deployment Descriptor Editor as shown in Figure 6-49.

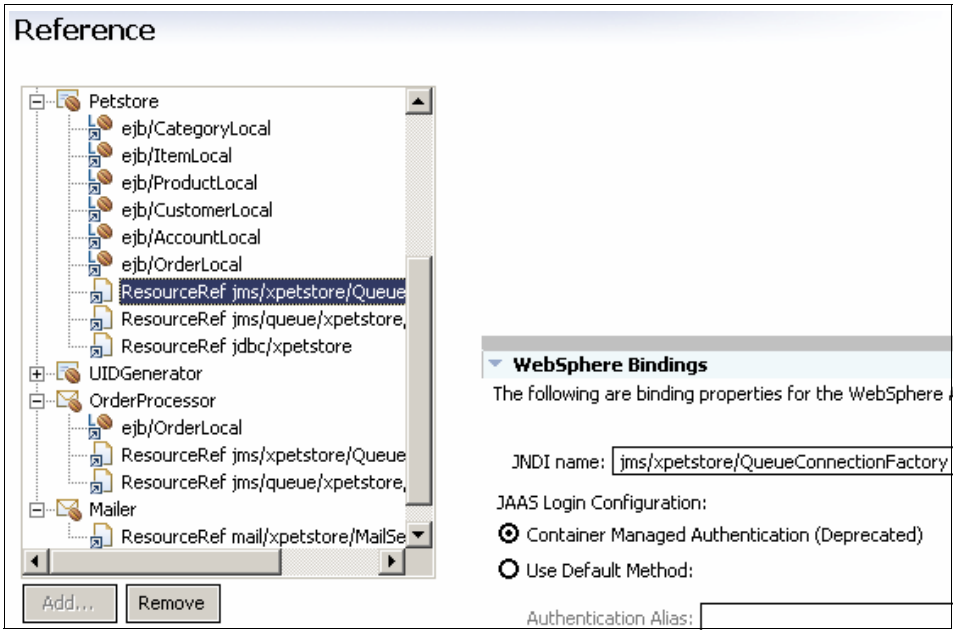


Figure 6-49 Resource reference page in the EJB Deployment Descriptor editor

2. Expand the **Petstore** bean by clicking the plus sign next to it. Select **ResourceRef jms/xpetstore/QueueConnectionFactory** from the list and specify the values shown in Table 6-47 the WebSphere Bindings section:

Table 6-47 Petstore EJB's WebSphere bindings

Parameter	Value
JNDI name	jms/xpetstore/QueueConnectionFactory

3. Select **ResourceRef jms/queue/xpetstore/order** from the list and specify the values shown in Table 6-48 in the WebSphere Bindings section.

Table 6-48 Order EJB's WebSphere bindings

Parameter	Value
JNDI name	jms/queue/order

4. Expand the **OrderProcessor** bean, select **ResourceRef jms/xpetstore/QueueConnectionFactory** and specify the values shown in Table 6-49 in the WebSphere Bindings section.

Table 6-49 OrderProcessor EJB's WebSphere bindings

Parameter	Value
JNDI name	jms/xpetstore/QueueConnectionFactory

5. Expand the **OrderProcessor** bean and select **ResourceRef jms/queue/xpetstore/mail** and specify the values shown in Table 6-50 in the WebSphere Bindings section.

Table 6-50 OrderProcessor EJB's WebSphere bindings

Parameter	Value
JNDI name	jms/queue/mail

6. Expand the **Mailer** bean and select **ResourceRef mail/xpetstore/MailSession** and specify the values shown in Table 6-51 in the WebSphere Bindings section.

Table 6-51 Mailer EJB's WebSphere bindings

Parameter	Value
JNDI name	mail/xpetstore/MailSession

Building the application

We used the Export wizard in Rational Application Developer V7.5 to create the build for the migrated xPetstore EJB.

1. Right-click the **xpetstore-ejb** EAR project and select **Export** from the menu.
2. From the Export context menu, select **EAR file** and click **Next**.
3. Specify a destination for the EAR file, select all the check boxes, and click **Finish**.

Deploying the application

We used the built-in WebSphere Application Server V7 test server in Rational Application Developer V7.5 to deploy the migrated application. However, you can use an external server to accomplish the same task.

1. In Rational Application Developer, Servers, view right-click **WebSphere Application Server v7.0 at localhost**. From the context menu select **Add and Remove Projects**. The **Add and Remove Projects** dialog box is displayed.
2. Under Available projects select **xpetstore-ejb** and click **Add**.

   The xpetstore-ejb project is under the **Configured projects**.
3. Click **Finish**. The application starts publishing on WebSphere.

Testing the application

Test the application as described in “Testing the application” on page 163. All test cases should pass and you should obtain the same test results as in the initial environment:

```
http://localhost:9080/xpetstore-test/index.html
```

You can also log in into the application and see how it works:

```
http://localhost:9080/xpetstore-ejb
```

6.6.4 Summary

While migrating the xPetstore EJB application, we were able to redeploy it from Oracle WebLogic Server 9.2 into the WebSphere Application Server V7 without changing much of the Java code. We had to update the deployment descriptors, including the mappings of CMP EJBs to the database. This is a sound statement for J2EE portability.

Lessons and best practices learned in this migration are discussed in more detail in Appendix A, “Development practices for portable applications” on page 325.

7

Migrating from JBoss

This chapter describes the migration of J2EE 1.4 and JEE 5 applications from JBoss to WebSphere Application Server V7, using the following three examples:

- JSF-EJB3 TODO application from JBOSS
- A simple JBoss Seam application generated with a seam-gen utility.
- Online brokerage sample application from Geronimo project v1.0, which was developed to have JBoss-specific features.

At the time of writing if this IBM Redbooks publication, the latest version of JBoss Application Server was 6.0.0.M2, which is a milestone release that features Java EE 6 technologies such as JPA 2.0 and Servlet 3.0. For the sake of specification level compatibility, we used the latest generally available version (which is 5.1.0, released in mid 2009). This version is an update on the Java EE 5 certified JBoss Application Server version 5.0.

This book has the following sections:

- “Migrating from older versions” on page 200
- “Preparing the environment” on page 200
- “Migrating JSF-EJB3 Sample Application from JBoss” on page 203
- “Migrating a Seam application generated with seam-gen” on page 226
- “Migrating the Online Brokerage application” on page 254

© Copyright IBM Corp. 2010. All rights reserved.

7.1 Migrating from older versions

IBM Redbooks publication *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690 describes how to migrate J2EE 1.3 applications from JBoss Application Server version 3.2.7 to WebSphere Application Server V6. If your application is at specification levels J2EE 1.3 and J2EE 1.4, and you are using XDoclet for code generation, you are advised to take a look at this version.

7.2 Preparing the environment

This section provides high-level steps and tasks for installing and configuring Apache Ant, Apache Maven, and JBoss Application Server. Instructions on how to install and configure WebSphere destination environment are covered in detail in Chapter 5, “Installation and configuration” on page 87.

7.2.1 Installing and configuring Apache Ant

Download Apache Ant binary distribution from the Apache Ant website:

`http://ant.apache.org/`

We used version 1.8.0 on our environment.

Find detailed installation instructions at the following web page:

`http://ant.apache.org/manual/install.html`

Follow the steps to install and configure Ant:

1. Extract the downloaded file to a suitable location. Use a short path name such as `C:\Ant`, as long filenames might cause problems due to OS restrictions. We used `C:\Ant` as the installation directory.
2. Set the environment variables by navigating to **Start** → **Control Panel** → **System**. Choose **Advanced** → **Environment Variables**. We want to use the same JRE as our integrated testing environment. So in our scenario, in the user variables section, we set JAVA_HOME variable as follows:

   ```
   JAVA_HOME=C:\Program Files\IBM\SDP\runtimes\base_v7\java
   ```

3. Set the ANT_HOME user variable to the folder that you specified in step 1. In our case we set is as follows:

   ```
   ANT_HOME=C:\Ant
   ```

4. Add Java and Ant executables to your path by appending the following code to your PATH user variable or creating a new one if it does not exist:

   ```
   PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin
   ```

5. Verify environment variables and Ant installation by opening a new command prompt and checking the outputs of the following commands:

   ```
   java -version
   ant -version
   ```

   Output shall present the current versions of the tools.

7.2.2 Installing and configuring Apache Maven

Download Apache Maven binary distribution from Apache Maven website:

http://maven.apache.org/

We used version 2.2.1 on our environment.

Find detailed installation instructions at the following web page

http://maven.apache.org/download.html

Perform the following steps to install and configure Apache Maven:

1. Extract the downloaded file to a suitable location. In this scenario, we used `C:\Maven` as the installation directory.
2. Set the environment variables by navigating to **Start** → **Control Panel** → **System**. Choose **Advanced** → **Environment Variables**. Set the M2_HOME and M2 user variables to the folder that you specified in step 1. In our case we set is as follows:

   ```
   M2_HOME=C:\Maven
   M2=%M2_HOME%\bin
   ```

3. Add a Maven executable to your path by appending the following code to your PATH user variable or creating a new one if it does not exist. At this stage, our PATH variable looks as follows:

   ```
   PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%M2%
   ```

4. Verify Maven installation by opening a new command prompt and checking the outputs of the following command:

   ```
   mvn -version
   ```

   Output shall present the current version of Maven.

7.2.3 Installing and configuring JBoss Application Server

The following describes JBoss Application Server installation instructions:

Installing JBoss Application Server Version 5.1.0

Download JBoss Application Server distribution from JBoss site using the following web page. As mentioned in introduction of this chapter, we used version 5.1.0.

http://www.jboss.org/jbossas/downloads

Detailed installation instructions for JBoss Application Server can be found at:

http://www.jboss.org/jbossas/docs/5-x/gettingstarted.html

Perform the following steps to install or configure JBoss Server:

1. Extract the downloaded file to a suitable location. We used `C:\jboss-5.1.0.GA` as the installation directory.
2. Set the environment variables by navigating to **Start-** → **Control Panel** → **System**. Choose **Advanced** → **Environment Variables**. Select JBOSS_HOME. This location is referred to as `<jboss_home>` throughout the document. In our case we set it as follows:

   ```
   JBOSS_HOME=C:\jboss-5.1.0.GA
   ```

3. Add JBoss executable to your path by appending following to your PATH user variable. At this stage, our PATH variable looks like:

   ```
   PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%M2%;%JBOSS_HOME%\bin
   ```

4. Because we are running under Java 6, copy the libraries in Example 7-1 from the `<jboss_home>/client` directory to the `<jboss_home>/lib/endorsed` directory.

Example 7-1 Libraries to be copied

```
jbossws-native-saaj.jar
jbossws-native-jaxrpc.jar
jbossws-native-jaxws.jar
jbossws-native-jaxws-ext.jar
```

5. Verify installation by starting the server with the **`run.bat`** command.

Installing JBoss Application Server Version 4.2.3

Download JBoss Application Server distribution from the JBoss site from the following web page:

http://www.jboss.org/jbossas/downloads

We need this version of JBoss application server to verify one of the sample application to be migrated in 7.3, “Migrating JSF-EJB3 Sample Application from JBoss” on page 203.

Extract the downloaded file to a suitable location. We used `C:\jboss-4.2.3.GA` as the installation directory. We refer to this location as `<jbossV4_home>` throughout the document.

7.2.4 JBoss Seam Framework

Download JBoss Seam Framework distribution from the following web page.

http://seamframework.org/Download

We used version 2.2.1.CR1 in our environment. Extract the downloaded file to a suitable location. We used `C:\jboss-seam-2.2.1.CR1` as the directory.

7.3 Migrating JSF-EJB3 Sample Application from JBoss

JSF-EJB3 is the sample application discussed in the article *Installation and Getting Started Guide for JBoss Application Server 5* on the following web page:

http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Installation_And_Getting_Started_Guide/5/html/Sample_JSF_EJB3_Application.html

This is a simple application that persists a list of TODO items to a database and provides means to create, update, and edit these entities.

We chose to migrate this application for the following reasons:

- Use of JPA, EJB3
- Use of JSF, JSF Expression Language
- Very simple. Good for demonstrating basic features of migration. We can demonstrate project scope changes without getting lost in the repetitive work that large projects might require for each module or component.
- Contains a J2EE 1.4 web module on which we can demonstrate Specifications Migration Wizard.

7.3.1 Migration approach

JSF-EJB3 application was built with non-IBM development tools, using Ant as a build tool. In this particular migration exercise, we also migrate the build environment to IBM Rational Application Developer for WebSphere Software, and use the build cycle that ships with this product.

The migration is performed in the following steps:

1. Verify that application works in its original destination environment. In this case we build, deploy, and run the application on JBoss Application Server 5.1.0.
2. Import the EAR file built for JBoss to IBM Rational Application Developer for WebSphere Software, to create the project structure. Import source files into projects created from source distribution.
3. Analyze and fix problems reported by the build environment and migration toolkit.
4. Check source code and configuration files with the Migration Tool.
5. Build the JSF-EJB3 sample application using IBM Rational Application Developer for WebSphere Software. Deploy and test the application on an integrated WebSphere Application Server V7 test environment.
6. (Optional) Specifications level migration.
7. (Optional) Migrate JPA provider to OpenJPA.

7.3.2 Verifying JSF-EJB3 sample application

The JSF-EJB3 application does not require any additional configuration on the target JBoss application server. The Application uses the default data source that is preconfigured on the application server.

Perform the following steps to verify JSF-EJB3 sample application:

1. Download sample application from the following web page:

   http://www.redhat.com/docs/manuals/jboss/jboss-eap-4.3/gettingstarted.zip

   Older versions of the Getting Started Guide tell that sample applications might be downloaded from the following web page:

   http://docs.jboss.org/jbossas/docs/Getting_Started_Guide/4/html/About_the_Example_Applications.html

Additional information about this sample application and the download URL can be found at the sample application wiki page:

http://community.jboss.org/wiki/SampleJSF-EJB3Application

2. Extract the downloaded archive file to a suitable location. In our case, we used `C:\gettingstarted`. Hereafter, this is referred to as `<jsfejb_home>`.
3. Open a command prompt and change to the directory containing the sample application code: `<jsfejb_home>\jsfejb3`.
4. Build and deploy the sample application by issuing the **`ant`** command in the `<jsfejb_home>\jsfejb3` folder.

   The build script terminates successfully with the message `BUILD SUCCESSFUL`.
5. Open another command prompt and start your JBoss Application Server version 4.2.3 by executing the **`<jboss4_home>/bin/run.bat`** command.
6. Copy the `jsfejb3.ear` file in `<jsfejb_home>\build\jars` folder to the `<jboss_home>\server\default\deploy` folder.
7. Point your browser to the following address and test the sample application generating, editing and deleting a couple of TODO items:

   http://localhost:8080/jsfejb3/

7.3.3 Importing the EAR file and source code to RAD

Perform the steps to import the EAR file and source code to RAD:

1. Launch RAD by navigating to **Start** → **IBM Software Delivery Platform** → **IBM Rational Application Developer 7.5** → **IBM Rational Application Developer**.
2. Launch the import wizard by navigating to **File** → **Import.** Expand the JavaEE node in the import source tree and select the EAR file.

3. In the Enterprise Application Import window, browse to the location of the EAR file generated by the ant build, which might be found at the following location:

   `<jsfejb_home>\jsfejb3\build\jars\jsfejb3.ear`

   See Figure 7-1. Click **Next** to continue.

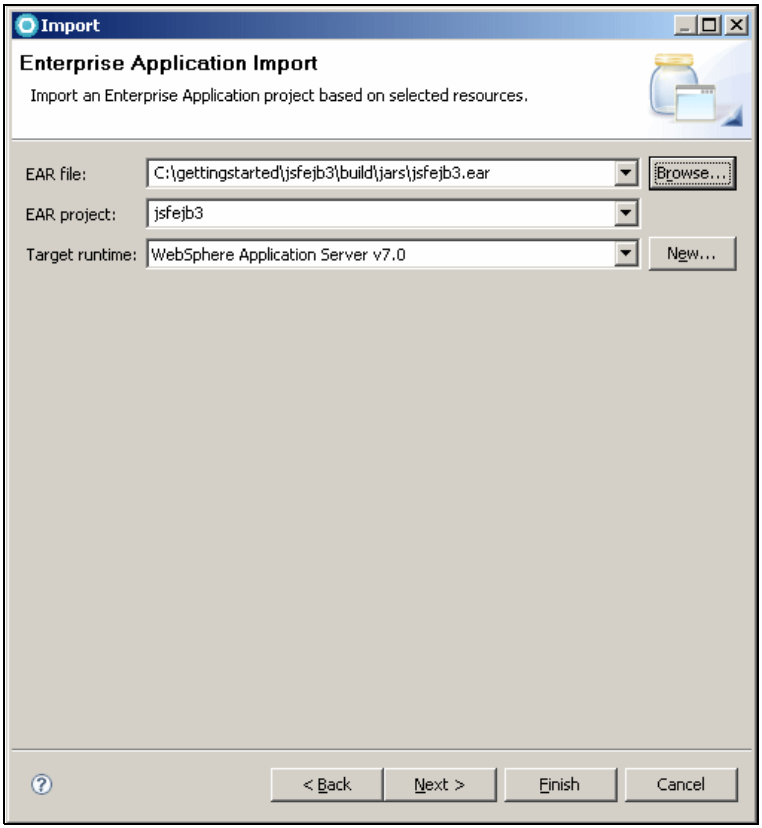


Figure 7-1 Importing an EAR file using import wizard

4. On the Enterprise Application Import window, click **Next** to continue without selecting `jsf-facelets.jar` to be imported as a utility project.

5. Click **Finish** on EAR Module and Utility JAR Projects window. The Import wizard runs and creates the necessary project and module structure. See Figure 7-2.

> **Configuring precompiled Java archive (JAR) libraries:** We do not have to configure precompiled Java archive (JAR) libraries for the web module or at the EAR level when we use the Import wizard to create the project and module structure. This is done automatically by the Import wizard when importing an EAR file.

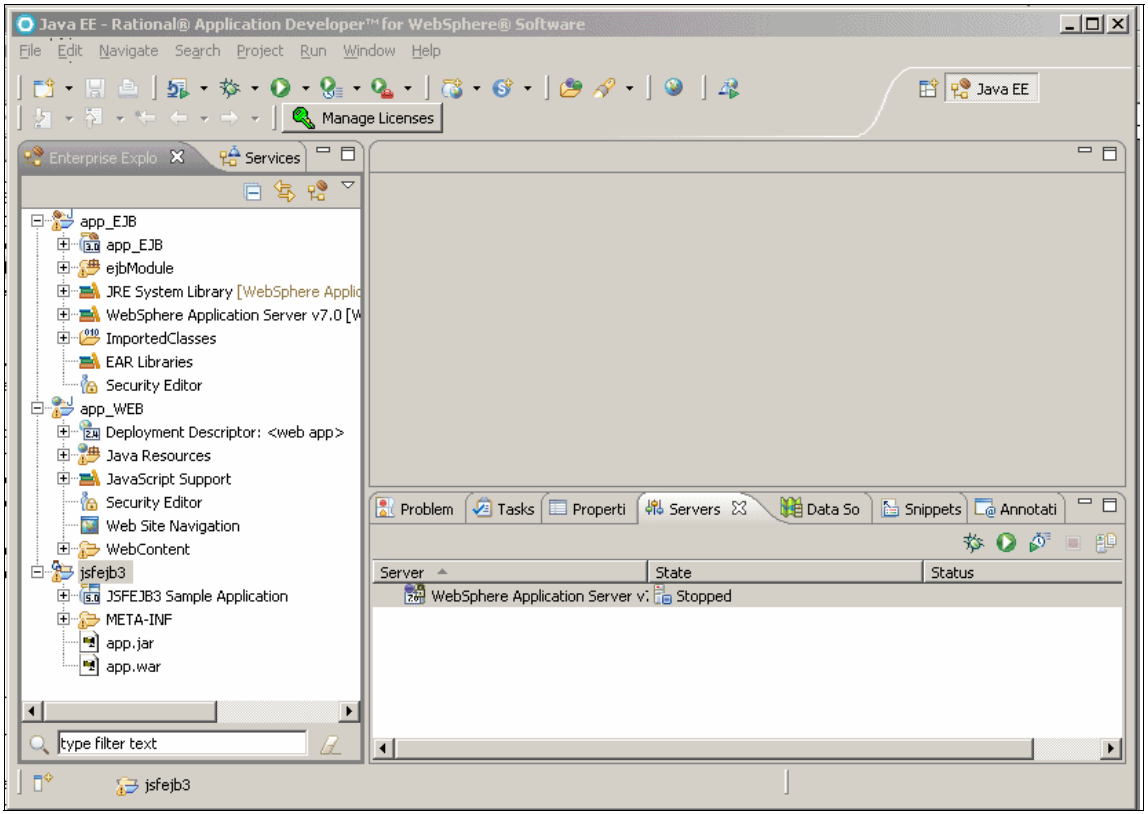


Figure 7-2 Initial project structure created by the Import wizard

6. Import the Java source code for the app_EJB module. Expand the project tree in the Enterprise Explorer view and investigate the Imported Classes node. This node lists packages that were imported in a compiled form.

Tip: While searching for classes to replace with their respective source files, the following rules apply:

- In a EJB module, classes might be found in the Imported Classes folder.
- In a web module, classes might be found under WEB-INF/classes. (While importing from JBoss, also check the WebContent node for classes).

7. The left side of Figure 7-3 shows the class files in our EJB module. We have three classes (in the `ImportedClasses` folder) in the default package. Import the respective source files for these classes as shown on the right side of Figure 7-3. Click and select **ejbModule**, then select **File** → **Import**. from the main menu.

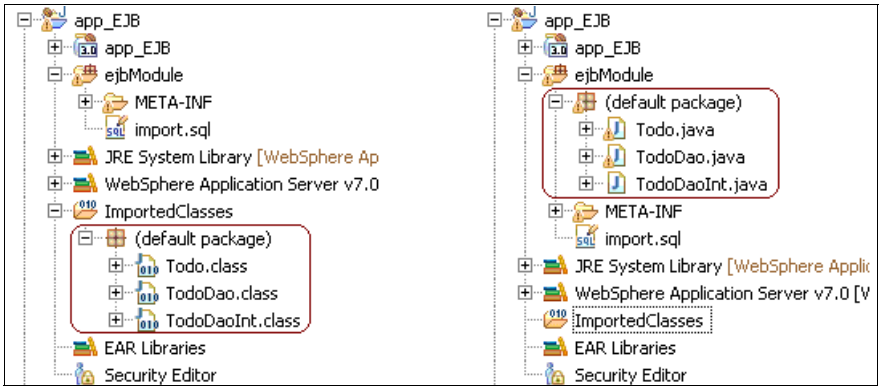


Figure 7-3 Before and after we replace class files with respective source files in EJB module

Note: While importing source files to replace classes imported by import wizard, following rules apply:

- In an EJB module, source files shall be added under ejbModule node.
- In a web module, source files shall be added under Java Resources/src node.

8. On the Import window, select **General** → **File System**. The File system import window is displayed. Click **Browse** and select the folder containing the source code for our project, `<jsfejb_home>\jsfejb3\src`.
9. Select the source files for the three classes in the `ImportedClasses` folder:
   - `Todo`
   - `TodoDao`
   - `TodoDaoInt`

   See Figure 7-4. Click **Finish**.

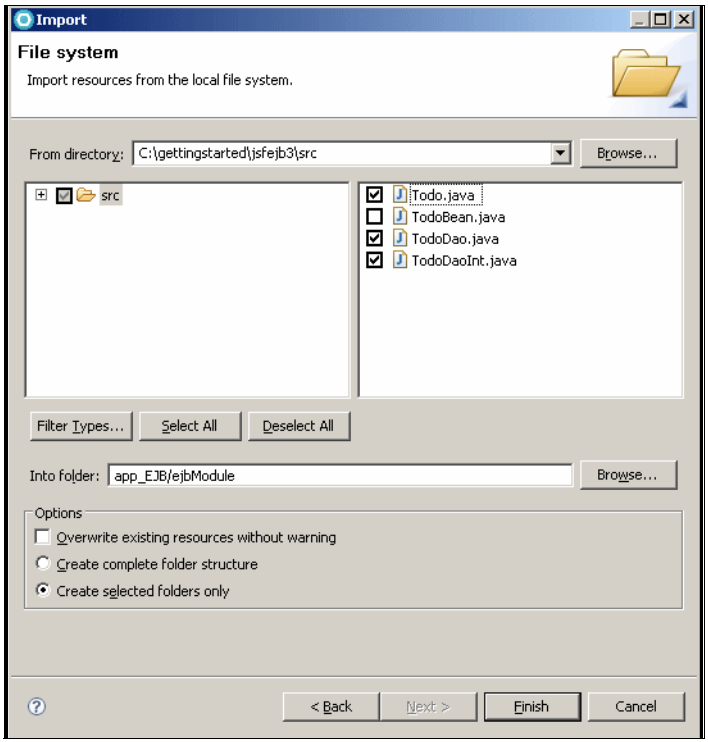


Figure 7-4 Importing source files for EJB module

10. Because we have imported source codes, we want to remove classes imported by the wizard in binary form. Click and select the default package in the ImportedClasses folder. Selecting **File** → **Delete** from the main menu displays a delete confirmation dialog box. Click **OK**.

    The EJB module, at this moment, looks like the right side of Figure 7-3 on page 208.

11. Import source code for the web module. As can be seen in Figure 7-5, we must replace `TodoBean` class with its source code. Click and select **Java Resources/src** under the web module, then select **File** → **Import** from the main menu.

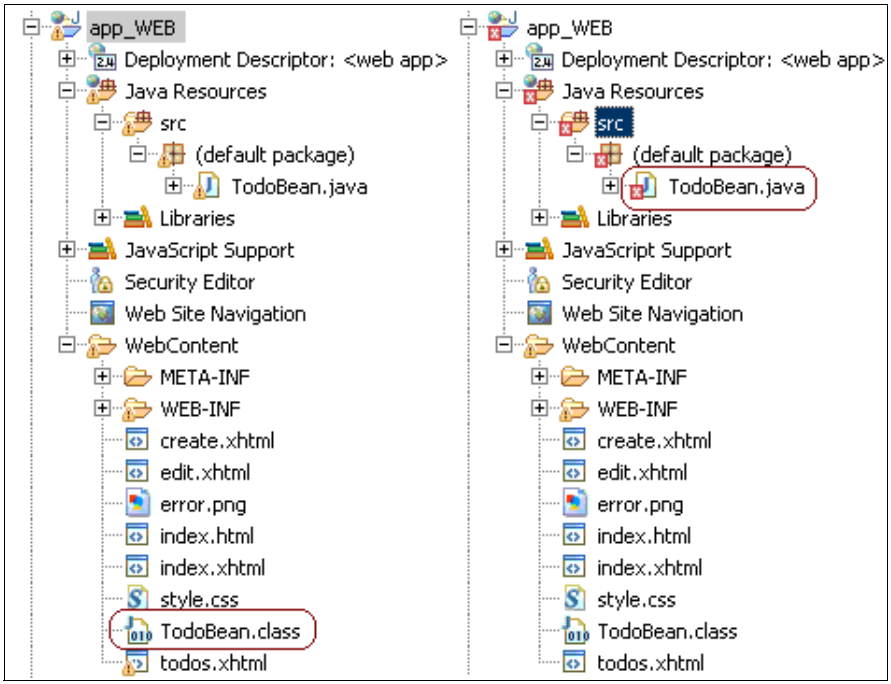


Figure 7-5 Before and after we replace class file with respective source file in web module

12. On the Import window, selecting **General** → **File System** displays the File System Import window. Click **Browse** and select the folder containing the source code of our project, `<jsfejb_home>\jsfejb3\src`.
13. Select the source file for the `TodoBean` class. Click **Finish**.
14. Click **TodoBean.class** under WebContent. Select **File** → **Delete** from the main menu. Click **OK** on the deletion confirmation dialog box that is displayed. The web module, at this moment, looks like the right hand side of Figure 7-5.

7.3.4 Analyzing and fixing migration problems

The following subsections describe the steps we performed for analyzing and fixing the problems.

Fixing the web module dependencies

As seen in Figure 7-5 on page 210, there is a compile-time problem in the newly imported `TodoBean.java` file, because it depends on the classes in the EJB module which are not yet visible. To fix this issue, perform the following steps:

1. Click the web module (app_WEB), and press Alt+Enter to open module properties dialog box.
2. Select **Java EE Module Dependencies** from left panel.
3. In the Java EE Module Dependencies view, select the app_EJB module check box, as can be seen at Figure 7-6.
4. Click **OK** to continue. As the workspace is rebuilt, compile error disappear.

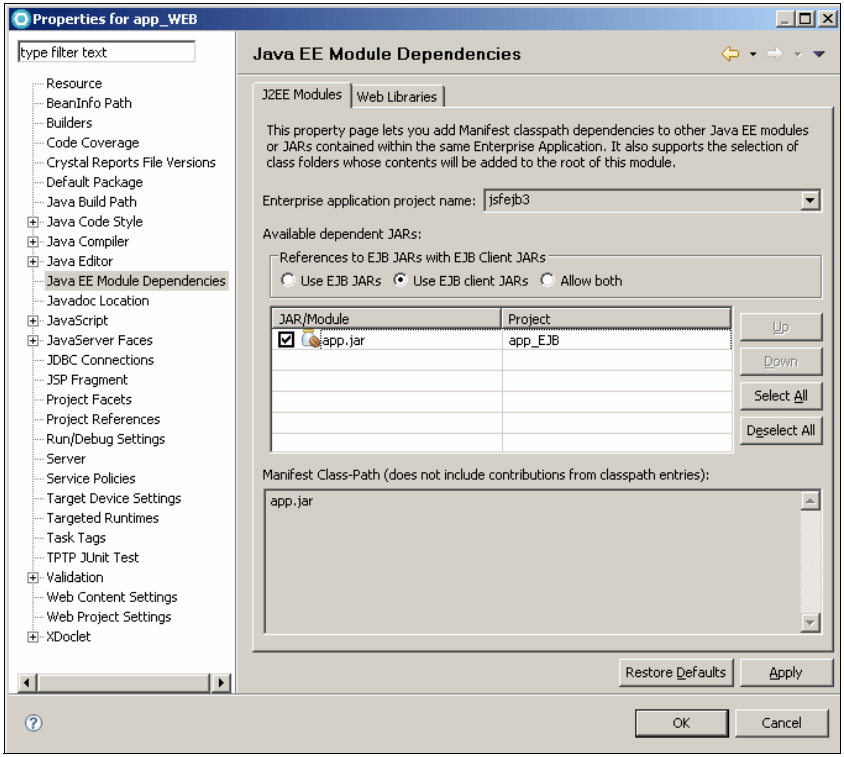


Figure 7-6 Adding a dependency to the web module

Fixing the JNDI lookup

TodoBean.java at the web module contains a JNDI lookup. Due to differences in default binding patterns, the JNDI name used in this lookup must be modified.

Table 7-1 summarizes default binding patterns for WebSphere Application Server V7. More information about default binding patters can be found at the following web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/cejb_bindingsejbfp.html

Table 7-1 WebSphere Application Server V7 default binding patterns

Description	Binding pattern
Short form local interfaces and homes	`ejblocal:<package.qualified.interface>`
Short form remote interfaces and homes	`<package.qualified.interface>`
Long form local interfaces and homes	`ejblocal:<component-id>#<package.qualified.interface>`
Long form remote interfaces and homes	`ejb/<component-id>#<package.qualified.interface>`

To fix the JNDI name in accordance with the aforementioned patterns, perform the following steps:

1. Modify `TodoBean.java` at the web module by replacing the `jsfejb3/TodoDao/local` JNDI name string with `ejblocal:TodoDaoInt`.
2. Save `TodoBean.java`.

7.3.5 Managing additional runtime dependencies

In the section we discuss the steps taken to manage runtime dependencies.

Fixing runtime dependencies

WebSphere Application Server V7 does not contain, by itself, certain libraries that are required to run the sample application. These are basically logging and hibernate libraries for persistence, and their dependencies. These libraries are included in JBoss Application Server distribution, so we can import these libraries into our application from JBoss Application Server installation.

There are several approaches on how to include these libraries, including packaging within the application archive file, or using shared libraries. For this sample application, we package external libraries into our application archive. Using shared libraries is demonstrated in the next migration samples.

Table 7-2 gives a list of libraries and from where to import them.

Table 7-2 Additional runtime libraries

Library	Location
antlr.jar	<jbossV4_home>\server\default\lib
cglib.jar	<jbossV4_home>\server\default\lib
concurrent.jar	<jbossV4_home>\lib
dom4j.jar	<jbossV4_home>\server\default\lib
hinernate3.jar	<jbossV4_home>\server\default\lib
hibernate-annotations.jar	<jbossV4_home>\server\default\lib
hibernate-entitymanager.jar	<jbossV4_home>\server\default\lib
javassist.jar	<jbossV4_home>\server\default\lib
jboss-common.jar	<jbossV4_home>\lib
log4j.jar	<jbossV4_home>\server\default\lib

We import these jar files into EAR module root folder by performing the following steps:

1. Select the top level jsfejb3 application module.
2. Launch the Import wizard by navigating to **File** → **Import** from the main menu.
3. On the Import window, select **General** → **File System** to display the File system import window. Click **Browse** and select the folder containing the libraries to be imported. Select the libraries and Click **Finish**.

4. Repeat steps 1 on page 213– 3 on page 213 for the two folders indicated in Table 7-2 on page 213. When this is complete, the application module looks like Figure 7-7.

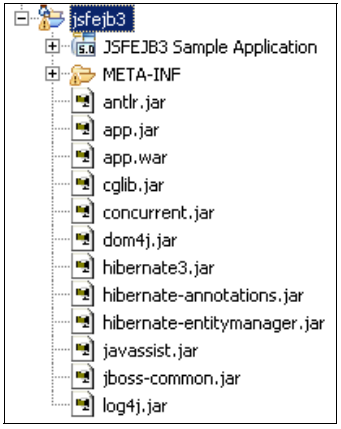


Figure 7-7 Application module after importing additional dependencies

5. Add dependencies to the EJB module to make the imported libraries available to this module. Click the **app_EJB** module and open the Properties window by pressing Ctrl+Enter.

6. Select **Java EE Module Dependencies** from list on the left. Select the check boxes next to each library that was imported. See Figure 7-8. Click **OK**.

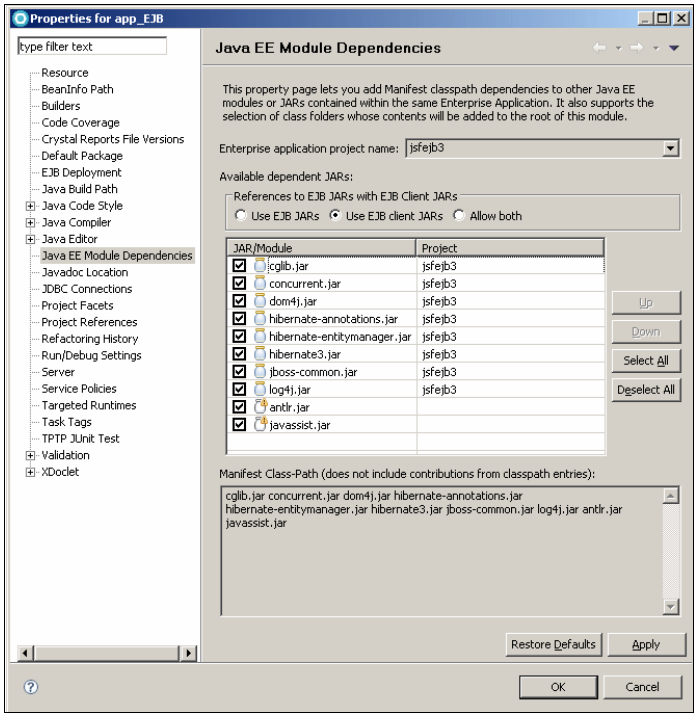


Figure 7-8 EJB module dependencies

Fixing persistence settings

Persistence settings are governed by the `persistence.xml` file in the `app_EJB\META-INF` folder. We need to make several modifications to this file, as outlined in following steps.

When you click **app_EJB\META-INF\persistence.xml**, the first thing you notice is that Persistence XML Editor is unable to open the file. It shows the warning message in Figure 7-9.

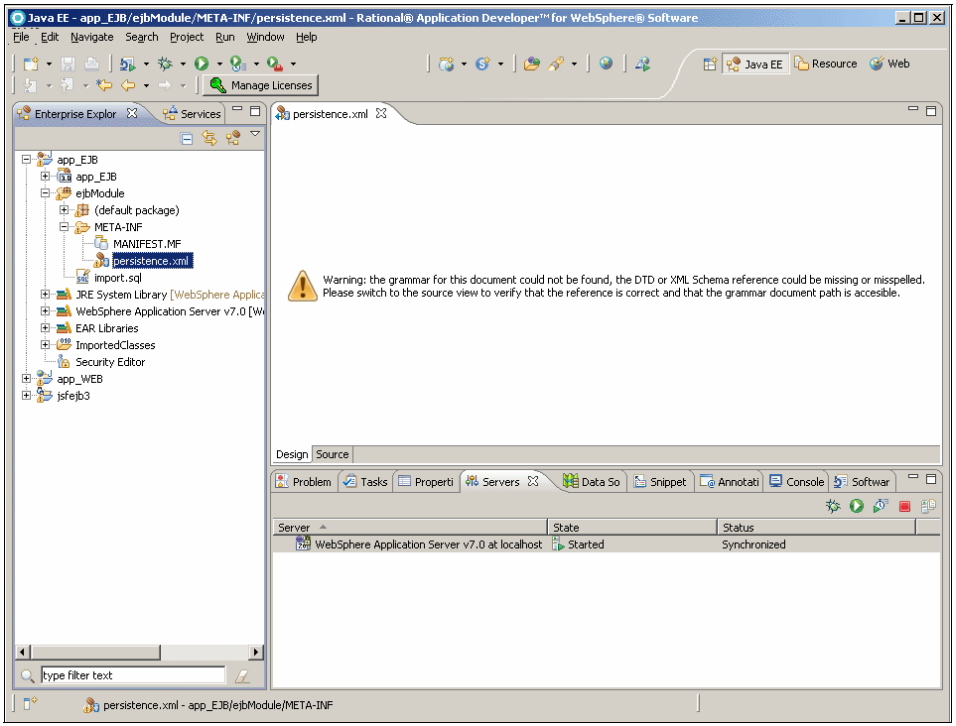


Figure 7-9 Persistence XML Editor warning message

We can continue editing by clicking the source tab of the editor and continue working. To fix this problem perform the following steps:

1. Switch to the source view of Persistence XML Editor.
2. Add grammar information to persistence.xml file by replacing the line containing `<persistence>` with the text in Example 7-2.

   Example 7-2 Replacement grammar information

   ```
   <persistence
      version="1.0"
      xmlns="http://java.sun.com/xml/ns/persistence"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
      http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
   ```

   From this point on, use either the graphical Persistence XML Editor, or continue working on the source tab. For simplicity's sake, we give instructions using source view.
3. Perform the following steps to use the WebSphere default data source that uses internal Derby provider:
   a. Change the name of the `jta-data-source` property from `java:/DefaultDS` to `DefaultDatasource`
   b. Change the value attribute of `hibernate.dialect` property from `org.hibernate.dialect.HSQLDialect` to `org.hibernate.dialect.DerbyDialect`.
4. Add following property to the properties list:

   ```
   <property name="hibernate.transaction.manager_lookup_class"
   value="org.hibernate.transaction.WebSphereExtendedJTATransactionLook
   up" />
   ```

Note: For more information about using Hibernate with WebSphere, see the developerWorks® article Using Spring and Hibernate with WebSphere Application Server at following web page:

http://www.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html

5. Two modifications on resources other than `persistence.xml` are necessary because of the behavior of DerbyDialect on identity columns in this version of Hibernate. Perform the following steps to make these modifications:

   a. In the source code of `Todo.java`, add an identity generation strategy by replacing the line `@Id @GeneratedValue` with `@Id @GeneratedValue(strategy=GenerationType.IDENTITY`.

      This change requires the import of the `javax.persistence.GenerationType` class. Add it manually at imports section or use the quick fix feature.

   b. Because the identity column is automatically generated, modify the insert statements in the `app_EJB\ejbModule\import.sql` file, which is used to populate database before first use. Modify each insert line by removing the ID column and value.

      The Insert statement shall look as follows:

      ```
      insert into Todo ( title, description) values ( 'This is a
      title', 'The description is here!')
      ```

6. Modify the application class loading mechanism. by performing the following steps:

   a. Click the **jsfejb3** application module. Right-click and select **Java EE** → **Open WebSphere Application Server Deployment**.

   b. Scroll down to Application section and change the Classloader mode field to **PARENT_LAST**. See Figure 7-10 on page 219.

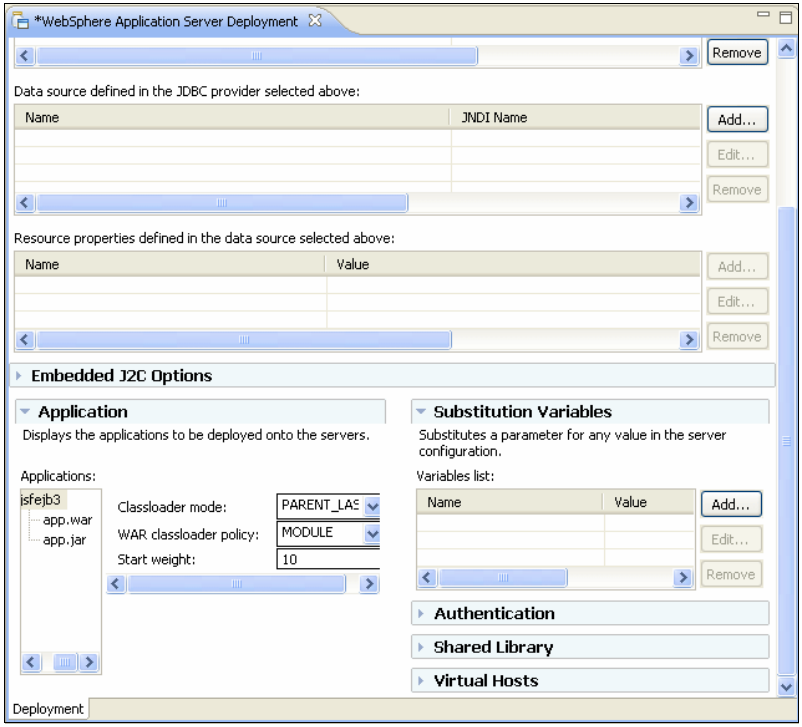


Figure 7-10 Configuring class loading for third party persistence providers

c. Click **File** → **Save** from main menu.

Note: This is done because we are using a third party persistence provider. More information about using third party persistence providers can be found at the following web page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tejb_jpa3rdparty.html`

7.3.6 Checking the project using Application Migration Tool

Even if we are working on a simple application, checking the source code and configuration files is a good idea. To run the Migration Tool on our project, perform the following steps:

1. Click the **jsfejb3** application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**.
2. On the left side of the Software Analyzer Configurations window, right-click **Software Analyzer**, and click **New**.
3. Click the Rules tab. Select **JBoss Application Migration** from the drop-down menu.
4. Click **Set** for the respective rules to become selected in the Analysis Rules and Domains area. See Figure 7-11.

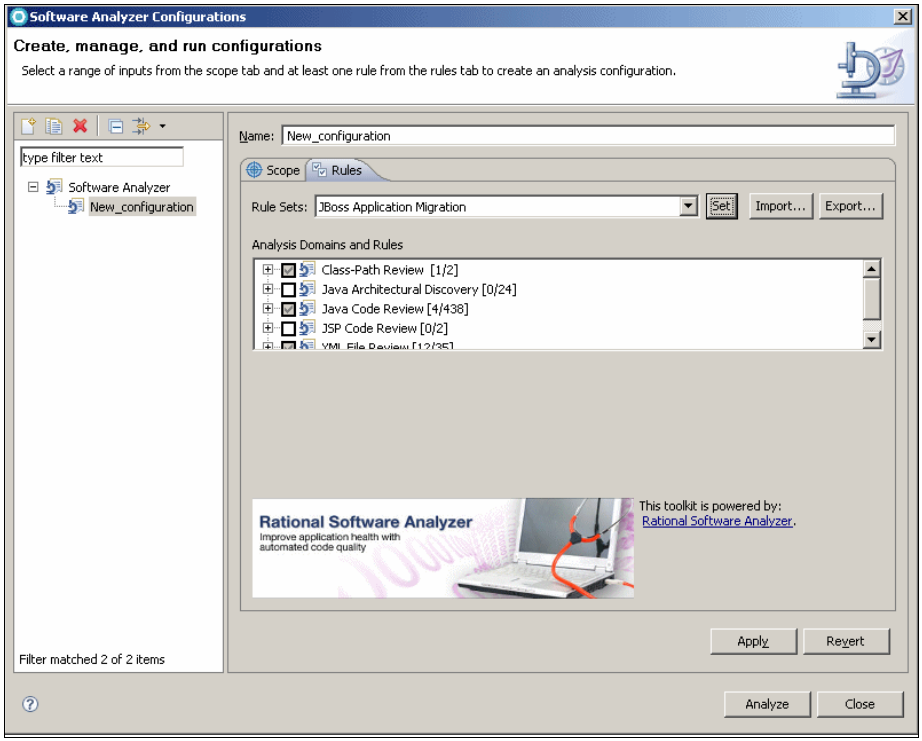


Figure 7-11 Software Analyzer configuration with JBoss migration rule set

5. Click **Analyze**. After the analysis is complete, the Software Analyzer Results view is displayed, docked to the bottom panel.

   In the Software Analyzer Results view, you can see a tab for each domain of analysis. In our case, we see a tab for Java Code Review, XML File Review, and Class-path Review.

6. Expand the XML file review section and the results in this section. Remove the `jsfejb3\META-INF\jboss-app.xml` file by **Delete**. See Figure 7-12.

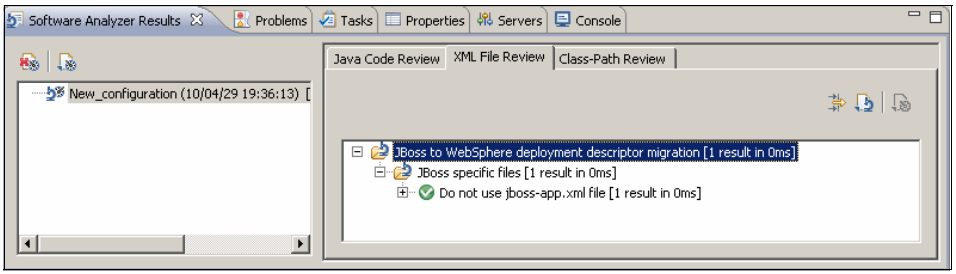


Figure 7-12 Migration Tool XML file review on JSFEJB3

7. Expand the class-path review section and the results in this section. See Figure 7-13.

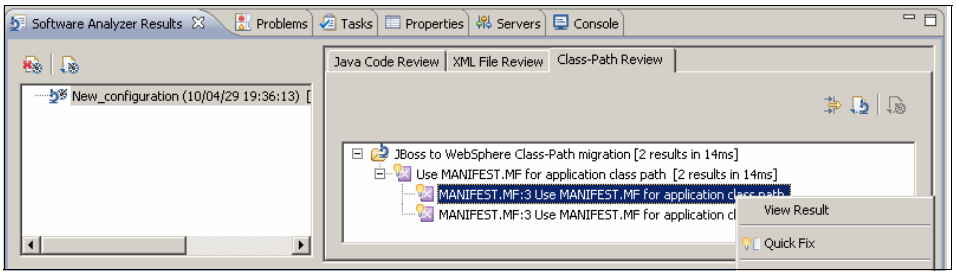


Figure 7-13 Migration Tool XML file review on JSFEJB3

8. Right-click one of the result lines and click **Quick fix all**.
9. Run the Software Analyzer again to ensure no more warnings are generated.

7.3.7 Build and run application on integrated test environment

Migration is complete and the application can be run on WebSphere Application Server V7. We can check how the migrated application works using integrated WebSphere Application Server V7 test environment.

To build, deploy, and run the application, perform the following steps:

1. Click the **app_WEB\WebContent\index.html** file. Right-click and select **Run As** → **Run** on the server. The Run on server dialog box is displayed. Keep the default settings, and click **Finish**.
2. After the application is deployed to test environment, an embedded web browser is displayed in the main docking area, presenting entry page of JSFEJB3 application. Follow the links to test the application functionality.

7.3.8 (Optional) Specifications level migration

IBM Rational Application Developer for WebSphere Software contains an integrated Specifications Migration Wizard that can migrate Java EE modules to higher specification levels.

More information about this topic can be found at the following web page:

`http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/index.jsp?topic=/com.ibm.etools.j2ee.doc/topics/tjmigrate.html`

The JSFEJB sample, at this point, is a Java EE 5 application that contains an J2EE web module, which is at level 2.4. We can automatically migrate this module to JEE 5 level (2.5) using integrated Specifications Migration Wizard.

Perform the following steps:

1. Right-click the **app_WEB** web module and select **Java EE** → **Specifications Migration Wizard**.

   A dialog box is displayed that prompts you to close all active editors. Click **Yes** to continue.

   The Welcome page is displayed. Read the contents and click **Next**.

   The Web project migration window is displayed, listing our web module as selected.

2. Select the target specification level using the J2EE version drop-down menu. See Figure 7-14. Because we are already at level 1.4, the only choice presented is 5.0. Click **Finish**.

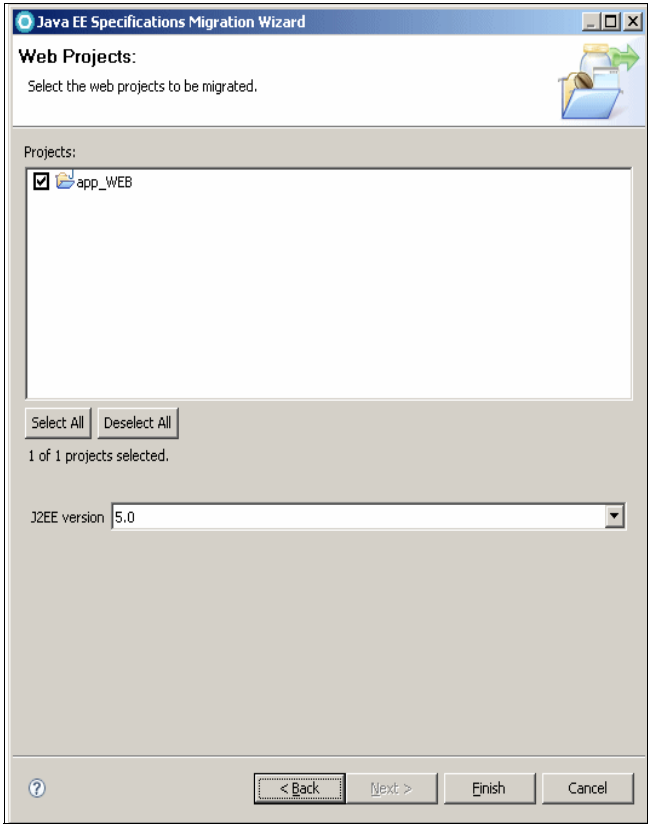


Figure 7-14 Target specification level selection

3. A summary dialog box (Figure 7-15) is displayed, which reports that migration was completed successfully.

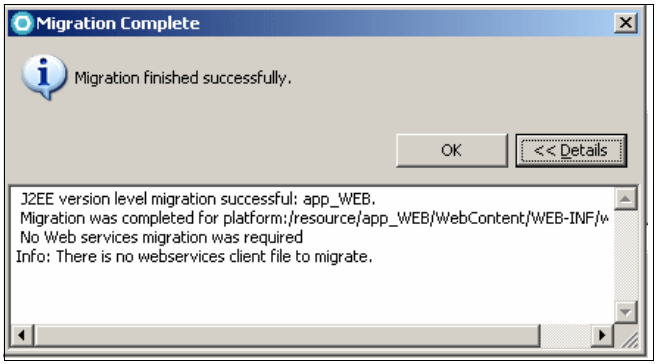


Figure 7-15 Specifications level migration summary

We have migrated our web module from Servlet specification version 2.4 to 2.5, as can be seen in Figure 7-16.

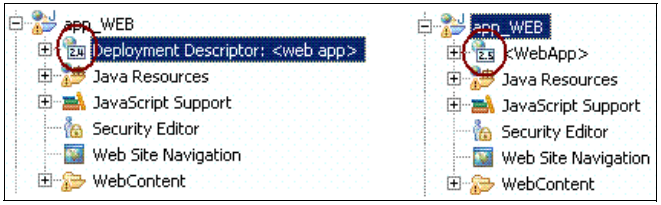


Figure 7-16 Before and after Specification Migration Wizard run

7.3.9 Migrating to WebSphere built-in JPA provider (optional)

Two persistence providers are included in WebSphere Application Server:

- JPA for WebSphere Application Server persistence provider
- Apache OpenJPA persistence provider

The JPA for WebSphere Application Server persistence provider is the default provider for WebSphere Application Server. You can use one of the two providers, or a third-party persistence provider, as the default provider.

If no JPA provider is configured in the <provider> element of the `persistence.xml` file within an EJB module, the default JPA provider that is currently configured for this server is used.

Roll back the classloading configuration changes we have performed for the third-party JPA provider and modify the `persistence.xml` file:

1. Click and select the jsfejb3 application module. Right-click, and select **Java EE → Open WebSphere Application Server Deployment**.
2. Scroll down to Application section and change Classloader mode to **PARENT_FIRST**, cancelling the change we performed before, as can be seen at Figure 7-10 on page 219.
3. Open the `persistence.xml` file in an editor by double-clicking it at `app_EJB/ejbModule/META-INF/persistence.xml`.
4. Remove the provider element. Also remove all property elements.
5. Add the following property element to the properties element:

   ```
   <property name="openjpa.jdbc.SynchronizeMappings"
   value="buildSchema(ForeignKeys=true)" />
   ```

   This property instructs OpenJPA to attempt to run the mapping tool on all persistent classes to synchronize their mappings and schema at runtime, creating tables at our database.
6. Save the `persistence.xml` file using **File → Save**.
7. To deploy the modified application to the server, click the Servers tab at the bottom. Right-click the server instance and select **Publish** from the context menu.
8. Click the `app_WEB\WebContent\index.html` file. Right-click and select **Run As → Run**. Test the application with the default persistence provider.

7.3.10 Summary

While migrating the JSF-EJB3 TODO sample application, we were able to demonstrate the following concepts:

- Source migration to Rational Application Developer for WebSphere Software as a build and test environment.
- Use of WebSphere Application Server Migration Tool v1.1 for Rational Application Developer. Note that JBoss migration support in this tool is limited at the time of writing, but increasing rapidly.
- Use of Rational Application Developer for WebSphere Software Specification Level Migration to migrate modules further to newer specification levels when imported.
- Migrating from third-party JPA providers to WebSphere Open JPA built-in provider.

The main points to highlight during this migration are as follows:

- Differences in default JNDI bindings require rework on JNDI lookups.
- Libraries that ship with JBoss Application Server distributions and not with WebSphere Application server create additional runtime dependencies that must be taken care of.
- Different built-in data sources used in this example made it necessary to modify persistence setting such as Hibernate dialect and key generation strategy.
- Using Hibernate for persistence requires transaction strategy configuration specific to WebSphere.
- Using a third-party JPA provider requires changes to the classloading configuration.

7.4 Migrating a Seam application generated with seam-gen

Seam is an application framework for Enterprise Java, which is widely used with JBoss Application Server. Because Seam 3 modules are at preview stage (alpha and beta releases) at the time of writing, we decided to use latest release from Seam 2. More information about Seam can be found at the following web page:

`http://seamframework.org/`

Seam distribution contains an utility called *seam-gen*, which helps to setup a Seam project from scratch. Script generates the project structure, including skeleton code. It also has the ability to generate a fully functional CRUD (create-read- update- delete) type application given a schema at a data source. We use this ability to generate our own simple application using a database schema with a single table.

We chose to migrate this application for the following reasons:

- Use of Seam framework, which is widely used with JBoss Application Server
- Project structure created with seam-gen is a representative of a well-designed Seam application structure. Because of this, seam-gen is probably being used to jump-start many similar real world applications as well.
- A simple, yet complete application, good for demonstrating and focusing on framework-specific constructs without getting into too much repetitive work for various modules and components.

7.4.1 Migration approach

In this migration exercise, we use IBM Rational Application Developer for WebSphere Software as the target development and build environment. Seam-gen utility also generates an Eclipse project that uses an Ant based build, which can be imported into Eclipse and IBM Rational Application Developer for WebSphere Software. Because this alternative is already provided, we demonstrate how we can use the build cycle of IBM Rational Application Developer for WebSphere Software instead.

The migration is performed in the following steps:

1. Create a simple database schema with a single table on DB2.
2. Use seam-gen to generate a Seam web application that does CRUD operations on this table.
3. Verify that the application works in its original destination environment. In this case we build, deploy, and run the application on JBoss Application Server 5.1.0.
4. Import the EAR file that is built for JBoss to IBM Rational Application Developer for WebSphere Software, to create the project structure. Import source files into projects created from source distribution.
5. Analyze and fix problems reported by the build environment and migration toolkit.
6. Check source code and configuration files with the migration tool.
7. Build a seam-gen sample application using IBM Rational Application Developer for WebSphere Software. Deploy and test the application in an integrated WebSphere Application Server V7 test environment.

7.4.2 Preparing database schema

We first create a new database user. Perform the following steps to do so:

1. Launch the Computer Management tool. Click **Start** and select **Settings** → **Control Panel** → **Administrative Tools** → **Computer Management**.
2. Select **Local Users and Groups,** and select **Users**.
3. Click **Action** → **New User** from the menu.
4. The New User window is displayed. Enter `dbuser` in the User name field. Enter `passwd` in the Password and Confirm password fields.

   Clear the **User must change password at next logon** option and select the **Password never expires** option.
5. Click **Create**, and click **Close** to return to the users list.

6. Select **dbuser** from user list. Right-click to open context menu and click **Properties**.
7. Click the Member Of tab.
8. Click **Add**.
9. Select the **DB2ADMNS** group.
10. Click **Apply**, and **OK**.

Next, we create a new database and a new table for this user:. Perform the following steps to do so:

1. Click **Start** and select **Programs** → **IBM DB2** → **DB2Copy1 (default)** → **Command Line Tools** → **Command Editor**.
2. Enter the code in Example 7-3 to the command area.

Example 7-3 Code snippet

```
CREATE DATABASE TESTDB;

CONNECT TO TESTDB USER DBUSER USING passwd;

CREATE TABLE "DBUSER"."BOOK" (
      "ISBN" VARCHAR(13) NOT NULL,
      "TITLE" VARCHAR(100) NOT NULL,
      "AUTHOR" VARCHAR(100) NOT NULL );

ALTER TABLE "DBUSER"."BOOK"
      ADD CONSTRAINT "PRIM_KEY" PRIMARY KEY
      ("ISBN");
```

3. Click **Execute**. A progress indicator is displayed. After the operations are completed, the results view displays the output, as in Figure 7-17.
4. Close the Command Editor.

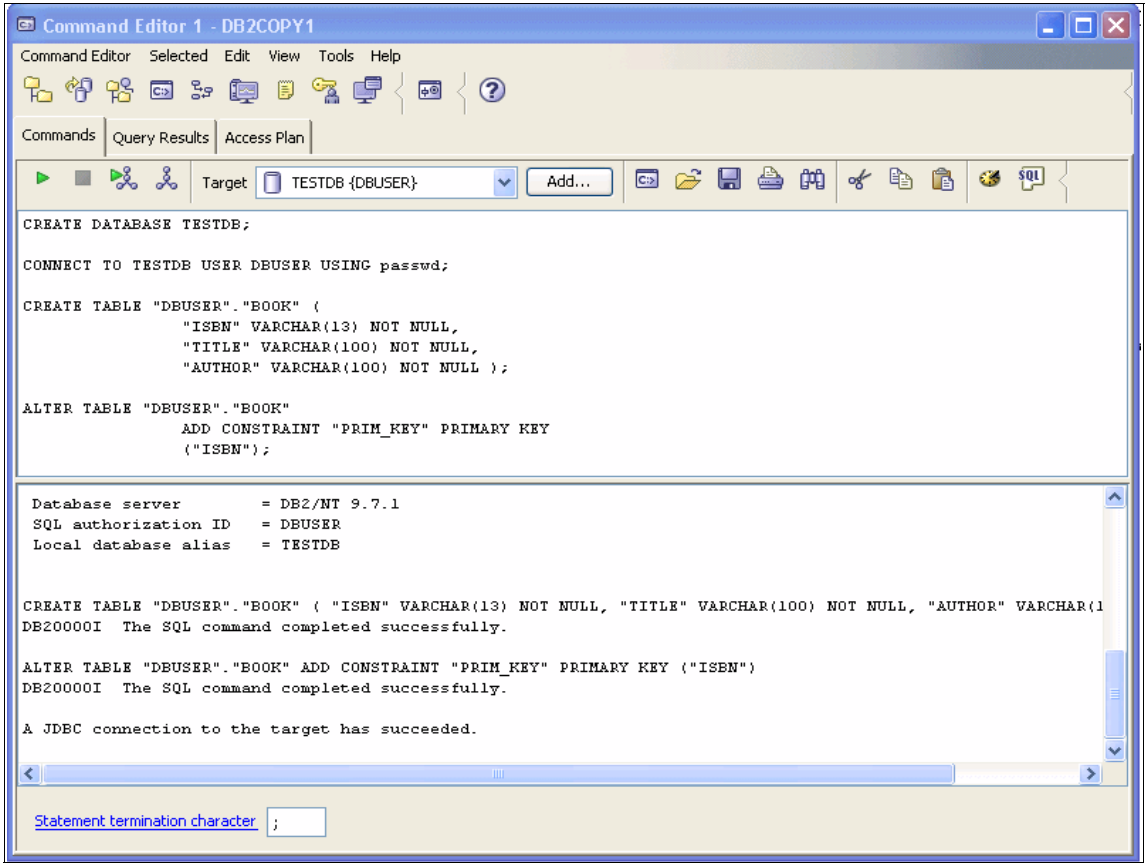


Figure 7-17 Creating test database using DB2 Command Editor

7.4.3 Generating Seam application using seam-gen

Next, we create a Seam web application that performs CRUD operations on the table we have just created. We begin by configuring seam-gen for our environment. Perform the following steps to do so:

1. Open a command prompt and change the directory to Seam installation folder, which, in our case, is `C:\jboss-seam-2.2.1.CR1`.
2. Type `seam setup` and press Enter. This begins an interactive script. See our responses in bold, shown in Example 7-4. Where you do not see a response, we pressed Enter to accept the default answer for the question.

Example 7-4 Seam setup command interactive output

```
C:\jboss-seam-2.2.1.CR1>seam setup
SEAM_HOME: C:\jboss-seam-2.2.1.CR1
Using seam-gen sources from: C:\jboss-seam-2.2.1.CR1\seam-gen
Buildfile: C:\jboss-seam-2.2.1.CR1\seam-gen\build.xml

init:

setup:
     [echo] Welcome to seam-gen 2.2.1.CR1 :-)
     [echo] Answer each question or hit ENTER to accept the default (in
brackets
)
     [echo]
    [input] Enter the directory where you want the project to be
created (should
 not contain spaces) [C:/Projects] [C:/Projects]

    [input] Enter your JBoss AS home directory [C:/Program
Files/jboss-5.1.0.GA]
 [C:/Program Files/jboss-5.1.0.GA]
C:\jboss-5.1.0.GA
    [input] Enter your JBoss AS domain [default] [default]

    [input] Enter your GlassFish V2 or V3 home directory (Ignore if you
aren't d
eploying to GlassFish) [C:/Program Files/glassfish-v2.1] [C:/Program
Files/glass
fish-v2.1]

    [input] Enter your GlassFish domain (Ignore if you aren't deploying
to Glass
```

```
Fish) [domain1] [domain1]

    [input] Enter the project name [myproject] [myproject]

     [echo] Accepted project name as: myproject
    [input] Do you want to use ICEfaces instead of RichFaces? [n] (y,
[n])

    [input] skipping input as property icefaces.home.new has already
been set.
    [input] Select a RichFaces skin [glassX] (blueSky, classic, darkX,
deepMarin
e, DEFAULT, emeraldTown, [glassX], japanCherry, laguna, ruby, wine)

    [input] Is this project deployed as an EAR (with EJB components) or
a WAR (w
ith no EJB support)? [war] (ear, [war])
ear
    [input] Enter the base package name for your Java classes
[com.mydomain.mypr
oject] [com.mydomain.myproject]

    [input] Enter the Java package name for your session beans
[com.mydomain.myp
roject.action] [com.mydomain.myproject.action]

    [input] Enter the Java package name for your entity beans
[com.mydomain.mypr
oject.model] [com.mydomain.myproject.model]

    [input] Enter the Java package name for your test cases
[com.mydomain.myproj
ect.test] [com.mydomain.myproject.test]

    [input] What kind of database are you using? [hsql] ([hsql], mysql,
derby, o
racle, postgres, mssql, db2, sybase, enterprisedb, h2)
db2
    [input] Enter the filesystem path to the JDBC driver jar [] []
C:/Program Files/IBM/SQLLIB/java/db2jcc.jar
    [input] Enter the filesystem path to the license jar [] []
C:/Program Files/IBM/SQLLIB/javadb2jcc_license_cu.jar
    [input] Enter the Hibernate dialect for your database
[org.hibernate.dialect
.DB2Dialect] [org.hibernate.dialect.DB2Dialect]
```

```
    [input] Enter the JDBC driver class for your database
[com.ibm.db2.jcc.DB2Dr
iver] [com.ibm.db2.jcc.DB2Driver]

    [input] Enter the JDBC DataSource class for your database
[com.ibm.db2.jcc.D
B2SimpleDataSource] [com.ibm.db2.jcc.DB2SimpleDataSource]

    [input] Enter the JDBC URL for your database [jdbc:db2:test]
[jdbc:db2:test]

jdbc:db2://localhost:50000/TESTDB
    [input] Enter the database username [sa] [sa]
dbuser
    [input] Enter the database password [] []
passwd
    [input] Enter the database schema name (Enter '-' to clear previous
value) [
] []
dbuser
    [input] Enter the database catalog name (Enter '-' to clear
previous value)
[] []

    [input] Are you working with tables that already exist in the
database? [n]
(y, [n])
y
    [input] Do you want to recreate the database tables and execute
import.sql e
ach time you deploy? [n] (y, [n])

[propertyfile] Creating new property file:
C:\jboss-seam-2.2.1.CR1\seam-gen\buil
d.properties
     [echo] Installing JDBC driver jar to JBoss AS
     [copy] Copying 1 file to C:\jboss-5.1.0.GA\server\default\lib

init:

init-properties:
     [echo] C:/jboss-5.1.0.GA

validate-workspace:
```

```
validate-project:

settings:
     [echo] JBoss AS home: C:/jboss-5.1.0.GA
     [echo] GlassFish home: C:/Program Files/glassfish-v2.1
     [echo] Project name: myproject
     [echo] Project location: C:/Projects/myproject
     [echo] Project type: ear
     [echo] IceFaces: n
     [echo] Action package: com.mydomain.myproject.action
     [echo] Model package: com.mydomain.myproject.model
     [echo] Test package: com.mydomain.myproject.test
     [echo] JDBC driver class: com.ibm.db2.jcc.DB2Driver
     [echo] JDBC DataSource class: com.ibm.db2.jcc.DB2SimpleDataSource
     [echo] Hibernate dialect: org.hibernate.dialect.DB2Dialect
     [echo] JDBC URL: jdbc:db2://localhost:50000/TESTDB
     [echo] Database username: dbuser
     [echo] Database password: passwd
     [echo]
     [echo] Type 'seam create-project' to create the new project

BUILD SUCCESSFUL
```

3. Enter the **seam new-project** command to generate a new project. Command ends with a BUILD SUCCESSFUL message.
4. Enter the **seam generate-entities** command to generate the components for our database table. Command ends with a BUILD SUCCESSFUL message.
5. Enter the **seam deploy** command to deploy the application to JBoss Application Server. Command ends with a BUILD SUCCESSFUL message.

   At this stage our application is packaged as an EAR file and has been copied to deployment directory of JBoss Application Server.

7.4.4 Verifying generated seam-gen application

Start JBoss Application Server (to which our sample application is already deployed) and use the application to perform operations on our table at the database, by performing the following steps:

1. Open a command prompt and change the directory to `<jboss_home>/bin` directory, In our case, this was `C:\jboss-5.1.0.GA\bin`.
2. Type the **`run.bat`** command to start the server.
3. After server launch process completes, launch a web browser and point the browser to `http://localhost:8080/myproject/`. The Seam application welcome window is displayed.
4. Select **Browse Data** → **Book List** from the Seam application menu.
5. As per default security configuration, browse and view the data without logging on. To add or modify data, log on to the application. When prompted, lo gin with admin credentials using `admin` as the username with an empty password.
6. Create, list, update, and delete several records to make sure the application is working without problems.

7.4.5 Importing the EAR file and source code to RAD

Follow these steps to import the EAR file and source code to RAD:

1. Launch Rational Application Developer for WebSphere Software using **Start → IBM Software Delivery Platform → IBM Rational Application Developer 7.5 → IBM Rational Application Developer**.
2. Launch the Import wizard using **File → Import.** Expand the **JavaEE** node in the import source tree and select **EAR file**. See Figure 7-18.

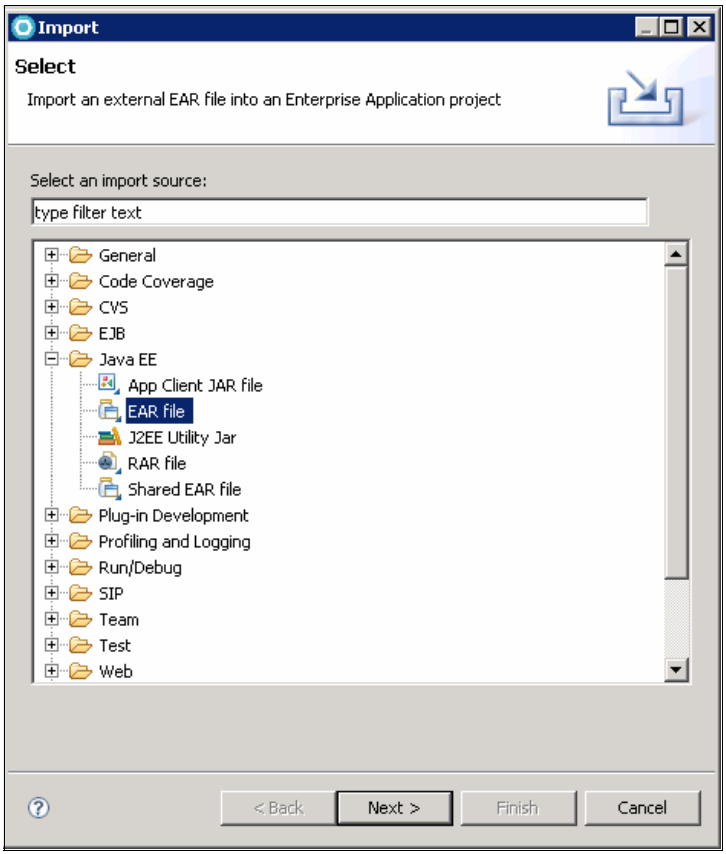


Figure 7-18 EAR import wizard

3. In the Enterprise Application Import window, browse to the location of the EAR file generated by seam-gen, which can be found at `C:\Projects\myproject\dist\myproject.ear`. See Figure 7-19. The location of this file is determined by the choices we made during seam setup script execution. Click **Next**.

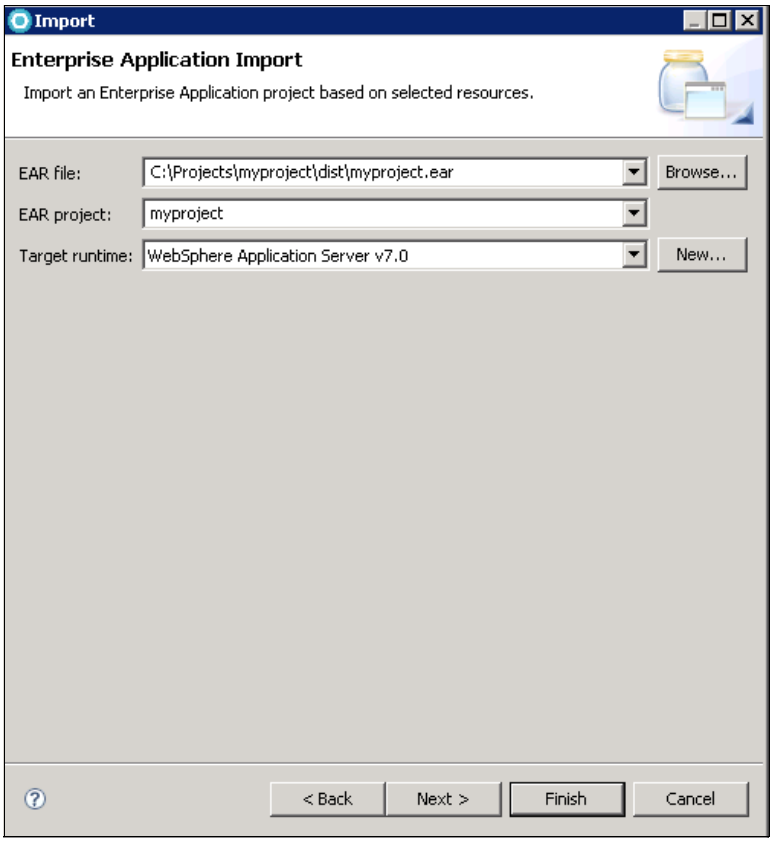


Figure 7-19 Importing myproject.ear

4. Click **Next** on the Utility jars and web libraries window.

5. On the EAR modules and utility jar projects window, clear the check boxes next to each library, except myproject.jar and myproject.war. See Figure 7-20. Click **Finish**.

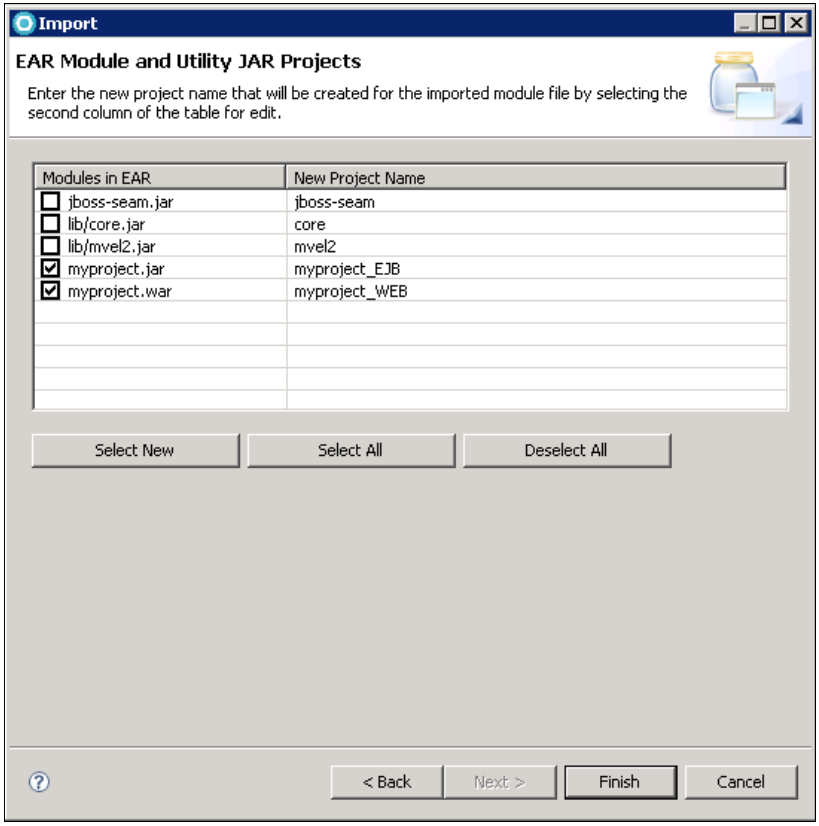


Figure 7-20 Selecting EAR Modules during import

6. Import source files to our project structure. Upon inspection we see that the myProject_EJB module has imported classes. Import the source files of these classes to the ejbModule folder and remove the class files from the project.

   Figure 7-21 shows the module structure before and after source code import.

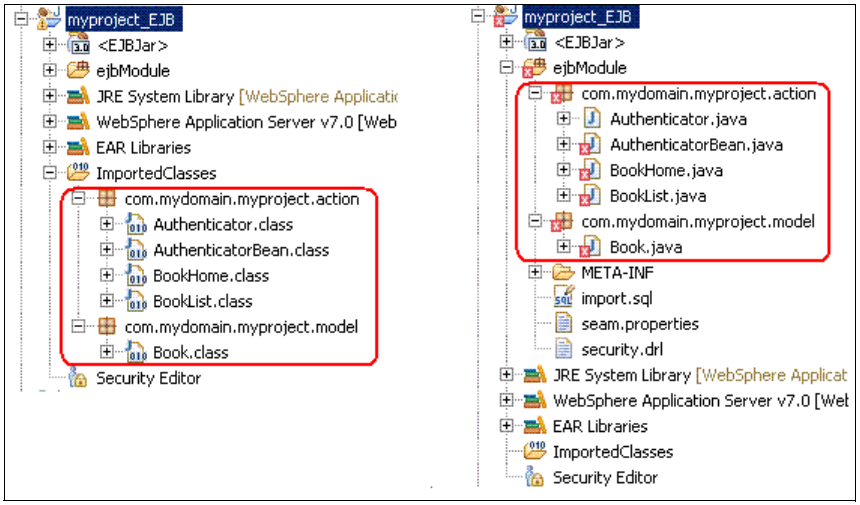


Figure 7-21 EJB module structure before and after source code import

7. Click the ejbModule node in the myproject_EJB module, and select **File** → **Import**.
8. Select **General** → **File System** in the Import window. The File System Import window is displayed. Import source code from two locations:
   - `C:\Projects\myproject\src\hot`
   - `C:\Projects\myproject\src\main`

9. Click **Browse** and select the `C:\Projects\myproject\src\hot` folder containing source code files for the com.mydomain.myproject.action package. Select all source files, as in Figure 7-22. Click **Finish**.

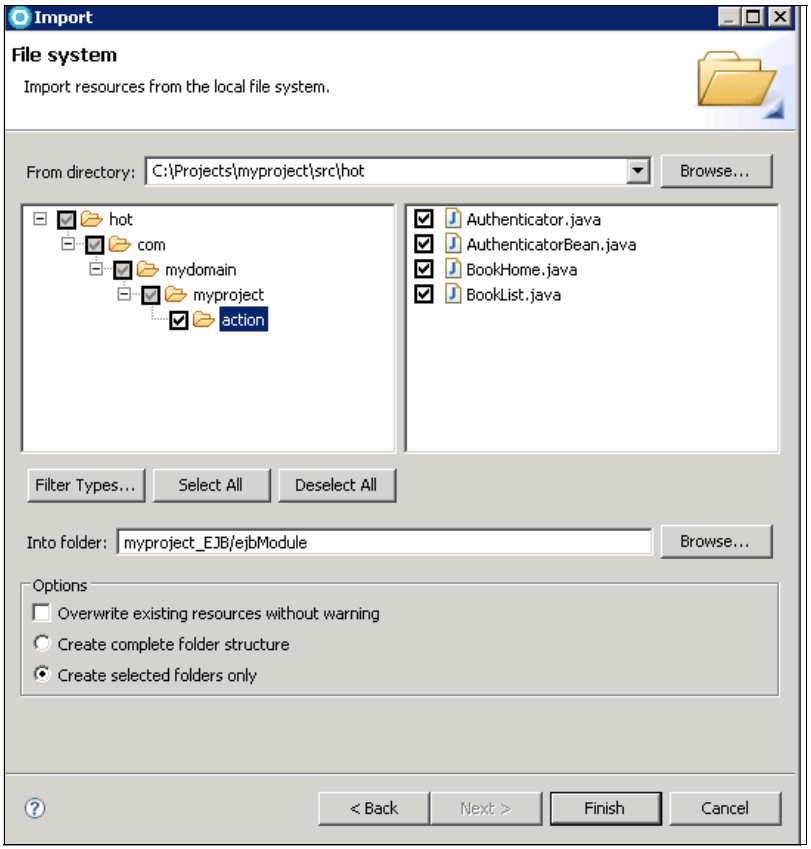


Figure 7-22 Importing source files

10. Repeat step 9 also for the `C:\Projects\myproject\src\main` folder, which contains source code for com.mydomain.myproject.model package.
11. Delete packages under the ImportedClasses node by selecting them, clicking **Delete**, and confirming deletion.

    At this point project structure looks like the right side of Figure 7-21 on page 238.

    Analyzing the web module, we can see that there are no imported classes that we must replace with source files. Resources such as html and xhtml files, stylesheets and images are imported automatically during EAR import.

7.4.6 Analyzing and fixing problems

The following sections discuss the steps taken to analyze and fix the migration problems.

Fixing compile time and runtime dependencies.

As can be seen on right side of Figure 7-21 on page 238, there are compile time errors on several classes (flagged with red cross icons). In our previous migration exercise, we included all of the necessary libraries for compile and runtime dependencies in EAR root to fix dependencies. When there are large number of libraries that can be reused by applications, a better approach might be to use shared libraries rather than libraries in packaged applications. This helps reduce the packaged size of the application, speeds up deployment, eases maintenance of libraries, and help to conserve server resources. We use shared libraries on this migration exercise:

1. Create a folder to contain the libraries. We chose `C:\sharedlib`. We refer to this location as `<shared_lib_dir>`.
2. As can be seen on left side of Figure 7-23, our EAR file already contains sixteen libraries that can be moved to a shared library. Select all of the libraries by holding the CTRL key.
3. Copy these libraries to the clipboard by right-clicking the selection to open context menu and selecting **Copy**.

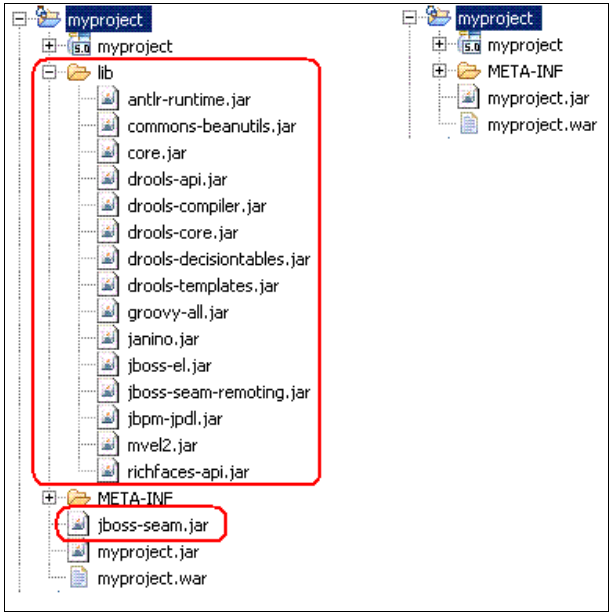


Figure 7-23 Before and after libraries in EAR moved to shared library

4. Paste the copied libraries to <shared_lib_dir>.
5. Delete the selected libraries from EAR module by selecting **Delete** from context menu. The result is the project structure shown on the right side of Figure 7-23 on page 240.

   Notice that we do not move myproject.war and myproject.ear. These files are refreshed as corresponding module projects are built.
6. Open the application deployment descriptor by clicking myproject/META-INF/application.xml.
7. In the Overview section of the Application Deployment Descriptor Editor, select the line EJB: jboss-seam.jar and click **Remove**.
8. Save your changes by selecting **File** → **Save** from main menu.
9. Select **Java EE** → **Update EAR Libraries** from the myproject module context menu.

   There are additional dependencies. These are the libraries that are shipped with JBoss Application Server, but are not a part of WebSphere Application Server distribution. These libraries are also included in our generated project, so we can copy them from our project. For our project, the additional dependencies in Table 7-3 must be copied to <shared_lib_dir> from C:\Projects\myproject\lib.

Table 7-3 .Shared dependencies to be copied

antlr.jar	bsh.jar	cglib-nodep.jar
commons-collections.jar	commons-logging.jar	concurrent.jar
dom4j.jar	gwt-servlet.jar	hibernate-annotations.jar
hibernate-commons-annotations.jar	hibernate-core.jar	hibernate-entitymanager.jar
hibernate-validator.jar	javassist.jar	jaxrs-api.jar
jboss-common-core.jar	jboss-seam.jar	jboss-seam-flex.jar
log4j.jar	myproject.jar	myproject.war
resteasy-jaxrs.jar	slf4j-api.jar	slf4j-log4j12.jar

10. Click the myproject application module. Right-click and select **Java EE** → **Open WebSphere Application Server Deployment** to display the WebSphere Application Server Deployment window.
11. Scroll down to Shared Library section. Expand the section and click **Add**.
12. Fill in the Name and Description fields. We used SeamSharedLibs as the shared library name.

13. Click **Browse** on the Class Path section of the window. A file selection dialog box is displayed (Figure 7-24). Using the dialog box, select **<shared_lib_folder>** and click **OK**.

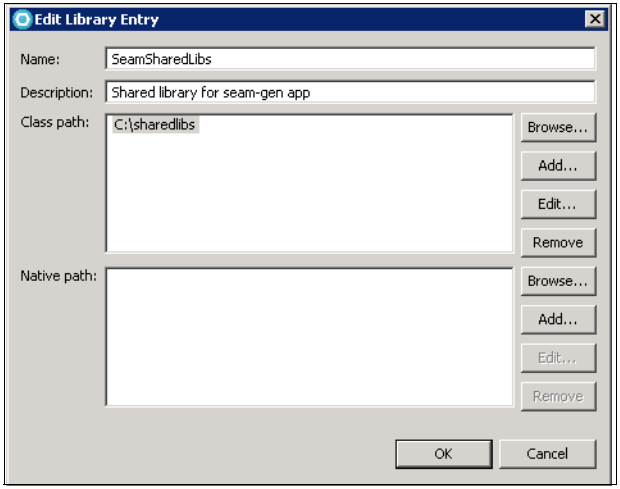


Figure 7-24 Shared library properties

We must modify the application class loading mechanism because we are using a third party persistence provider. More information about using third party persistence providers might be found at the following web page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tejb_jpa3rdparty.html`

14. Scroll down to Application section and change the Classloader mode to **PARENT_LAST.**
15. Type CTRL+S to save your changes.
16. While shared library resolves runtime dependencies, the myproject_EJB module also has compile time dependencies on the libraries we moved to the shared library. To resolve the compile time dependencies, select the myproject_EJB module, right-click and select **Build Path** → **Configure Build Path**.
17. Select the `Libraries` tab in the Java Build Path window. Click **Add External JARs.** A file selection dialog box is displayed.
18. Navigate to `<shared_lib_folder>` and select the following libraries to add to build path:
    - `jboss-seam.jar`
    - `hibernate-validator.jar`

19. Click **OK** to continue. The class path looks like Figure 7-25.

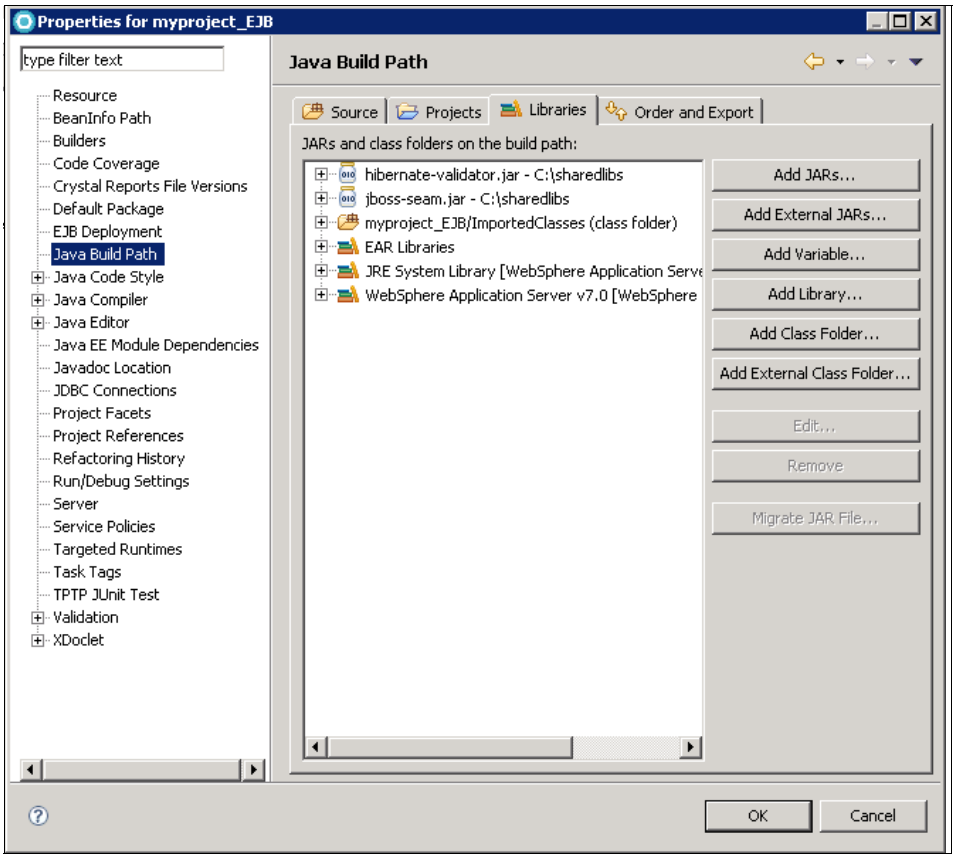


Figure 7-25 EJB module class path

20. Select **Project** → **Clean** to rebuild the project. No compile time errors remain.

Fixing JNDI lookup

In the previous migration exercise it was enough to change JNDI lookups in the source code to reflect the naming scheme. When using Seam however, we must find a more elaborate solution, because the Seam framework itself does JNDI lookups to make injections. Configuring Seam for WebSphere JNDI name space is documented with several alternatives at Seam documentation at the following web page:

`http://docs.jboss.org/seam/2.2.1.CR1/reference/en-US/html/websphere.html`

For our project, we use strategy 1, as explained on the aforementioned document. We add @JndiName annotation to session beans, create a seam-specific properties file (the content of which is specified in the Seam documentation), change JNDI matching pattern at `components.xml`, and add an EJB reference for LocalEJBSynchronizations. Perform the following steps to accomplish these tasks:

1. Start by annotating session beans with the @JndiName annotation. Our single session bean is AuthenticatorBean at `myproject_EJB/ejbModule/com.mydomain.myproject.action.AuthenticatorBean.java`. Open this file in the editor.
2. Add the following line between the @Name annotation and the class declaration line:

   ```
   @JndiName("ejblocal:com.mydomain.myproject.action.Authenticator")
   ```

   Note: JndiName uses the name of the localinterface class instead of bean class name.

3. This annotation requires the org.jboss.seam.annotations.JndiName class to be added to imports. Use quick fix on annotation by pressing CTRL+1 on the @JndiName annotation and clicking the I**mport JndiName** option.
4. Open the `myproject_WEB/WebContent/WEB-INF/components.xml` file in the editor.

   Replace `<core:init debug="@debug@" jndi-pattern="@jndiPattern@"/>` with `<core:init debug="false" jndi-pattern="java:comp/env/#{ejbName}"/>`
5. Create a new file named `seam-jndi.properties` at `myproject_WEB/Java Resources/src`. Content of this file is described in the Seam documentation. Copy the contents from the documentation to the generated file and save your changes.

   Tip: Seam documentation states that the `seam-jndi.properties` file shall be created at `WEB-INF/classes/seam-jndi.properties` in the web module. Because this folder is not visible in Rational Application Developer for WebSphere Software, use the src folder, because non-Java resources in this folder are copied to the classes folder during build.

6. Open the `myproject_WEB/WebContent/WEB-INF/web.xml` file in the editor. Switch to the source tab for easier editing.

7. Add the snippet in Example 7-5 inside the web-app element as the last element and save your changes.

Example 7-5 Code snippet

```
<ejb-local-ref>
   <description></description>
   <ejb-ref-name>EjbSynchronizations</ejb-ref-name>
   <ejb-ref-type>Session</ejb-ref-type>
   <local-home></local-home>
   <local>org.jboss.seam.transaction.LocalEjbSynchronizations</local>
   <ejb-link>EjbSynchronizations</ejb-link>
</ejb-local-ref>
```

Fixing the persistence configuration

The web module deployment descriptor, by generation, contains a persistence-unit-ref to the persistence unit definition at the EJB module. The indirect naming scheme used in the persistence-unit-name element of this reference does not work with WebSphere. Because of this we move `persistence.xml` to web module and use the jar-file element inside `persistence.xml` to reference beans at the EJB module. Also, we configure Hibernate properties for WebSphere as demonstrated in the previous migration exercise:

1. Select the `persistence.xml` and `orm.xml` files in the `myprojectEJB/ejbModule/META-INF` directory.
2. Right-click to open the context menu and select **Move**.
3. In the destination area, select the `myProject_WEB/WebContent/META-inf` folder and click **OK**.
4. Select myproject_Web and right-click to open the context menu. Select **Properties** → **Java EE module dependencies**.

5. Select the myproject_EJB check box as shown in Figure 7-26 and click **OK** to continue.

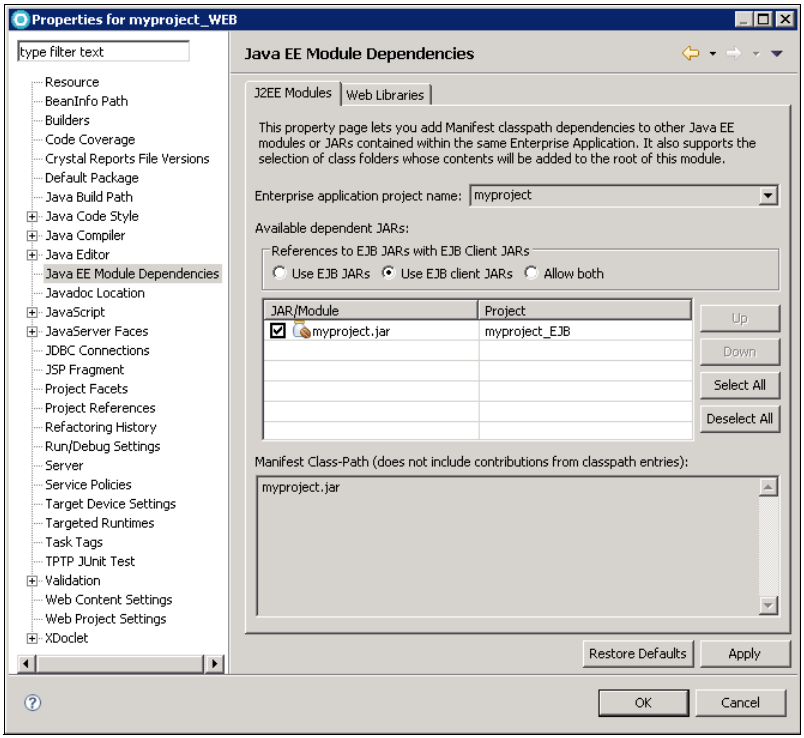


Figure 7-26 Web module dependencies

6. Open the `persistence.xml` file copied in the editor. Switch to the source tab for faster editing.
7. Add the following snippet to `persistence.xml`, beneath the <jta-data-source> element:

   ```
   <jar-file>myproject.jar</jar-file>
   ```

8. Remove the following line from `persistence.xml`:

   ```
   <property name="jboss.entity.manager.factory.jndi.name"
   value="java:/myprojectEntityManagerFactory"/>
   ```

9. Add the following line inside the <properties> element in `persistence.xml`:

   ```
   <property name="hibernate.transaction.manager_lookup_class"
   value="org.hibernate.transaction.WebSphereExtendedJTATransactionLook
   up" />
   ```

10. Type CTRL+S to save your changes.

11. Fix the persistence unit reference in the web module. Open `myproject_WEB/WebContent/WEB-INF/web.xml` in the editor. Switch to the source tab.
12. Replace the line `<persistence-unit-name>../myproject.jar#myproject</persistence-unit-name>` with `<persistence-unit-name>myproject</persistence-unit-name>`.

Creating the data source

As seen in `persistence.xml`, persistence configuration depends upon the existence of a data source with the myprojectDatasource JNDI name. Creating this data source for the test environment can be done in Rational Application Developer for WebSphere Software. Making configuration changes on a normal WebSphere Application Server installation through the administrative console is similar.

1. Select the myproject application module and right-click to open the context menu. Select **Java EE** → **Open WebSphere Application Server Deployment.**
2. Scroll down to the Authentication section. Click **Add** near the JAAS Authentication list subsection. The Add JAAS Authentication Entry dialog box is displayed.
3. Enter `db2user` for the alias, user ID, and description fields, and `passwd` for the password field. Click **OK**. At this point new authentication alias is listed in the JAAS Authentication list.
4. Scroll up to the Data Sources section, click **Add** near the JDBC provider list subsection.
5. Select `IBM DB2` as the database type and `DB2 Universal JDBC Driver Provider (XA)` as the JDBC provider type in the Create JDBC Provider window. Click **Next**.
6. Enter `DB2 JDBC Provider (XA)` in the name field of the JDBC Provider Details window.
7. Select the entries in the classpath section and click **Remove**.

8. Click **Add External JARs**. A file selection dialog box is displayed. Navigate to the `C:\Program Files\IBM\SQLLIB\java` folder and select the following JARs and click **OK** to add.

   - `db2jcc.jar`
   - `db2jcc_license_cu.jar`

   The JDBC provider settings window (Figure 7-27) is displayed. Click **Finish**.

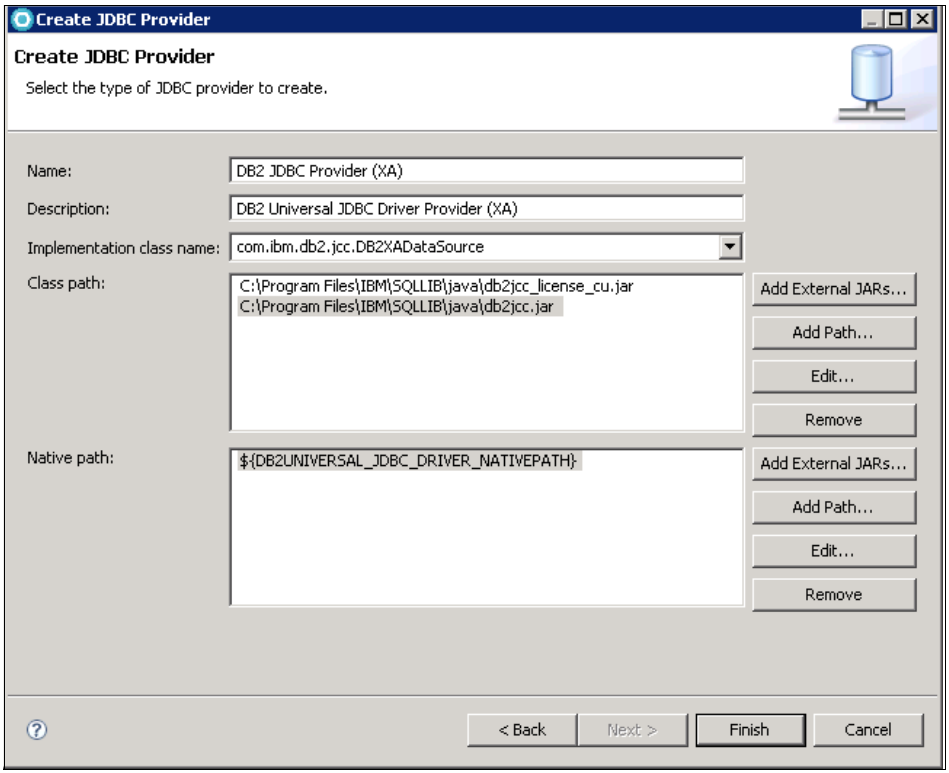


Figure 7-27 JDBC provider settings

9. The new JDBC provider is listed in JDBC providers list. Click this line to select **DB2 JDBC Provider (XA)**, and then click **Add** in the data sources subsection.

10. Select **DB2 Universal JDBC Driver Provider (XA)** in the Create Data Source window (Figure 7-28). Click **Next**. The Data Source details window is displayed.

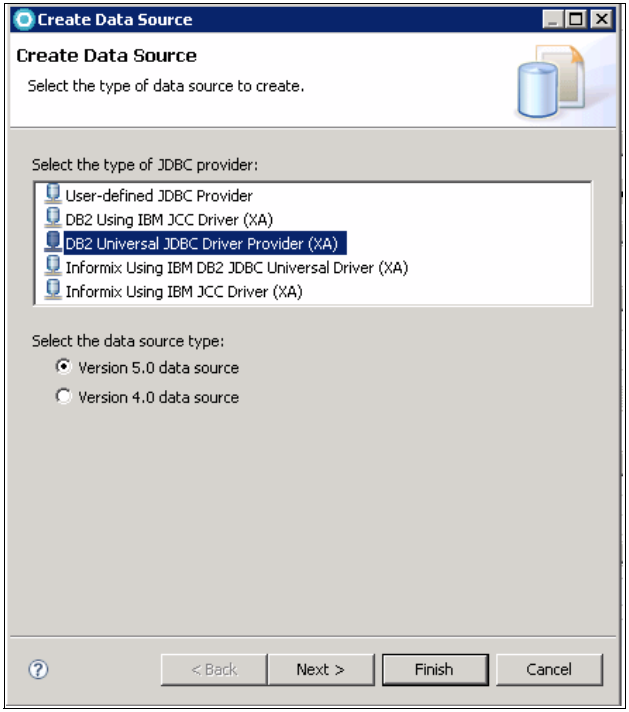


Figure 7-28 Selecting provider for data source

11. In the JNDI Name field, enter `myprojectDatasource`. Remember that this value should match the data source name specified in the `persistence.xml` file.

12. In the Container-managed authentication alias and Component-managed authentication alias fields, select the **dbuser** authentication alias that we created before using combo box. The window looks like Figure 7-29. Click **Next**. The Create Resource Properties window is displayed.

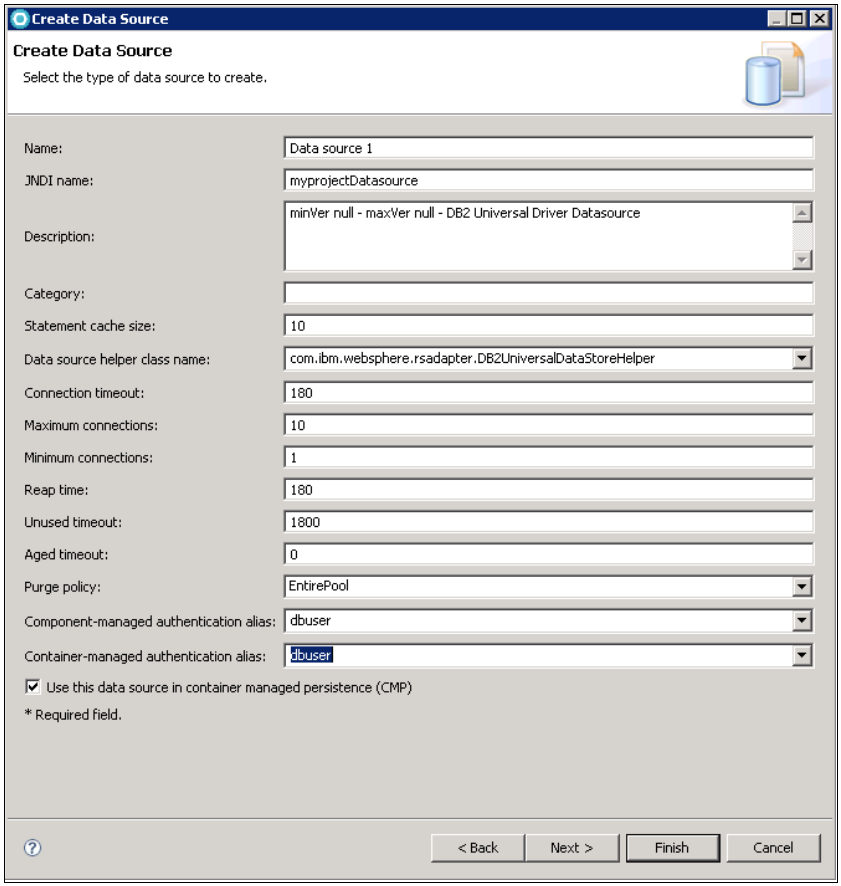


Figure 7-29 Data Source details

13. In the Create Resource Properties window (Figure 7-30), select the databaseName Resource Property. In the value field for this property, enter `testdb`. Select the serverName property. In the value field for this property, enter `localhost`. Click **Finish**.

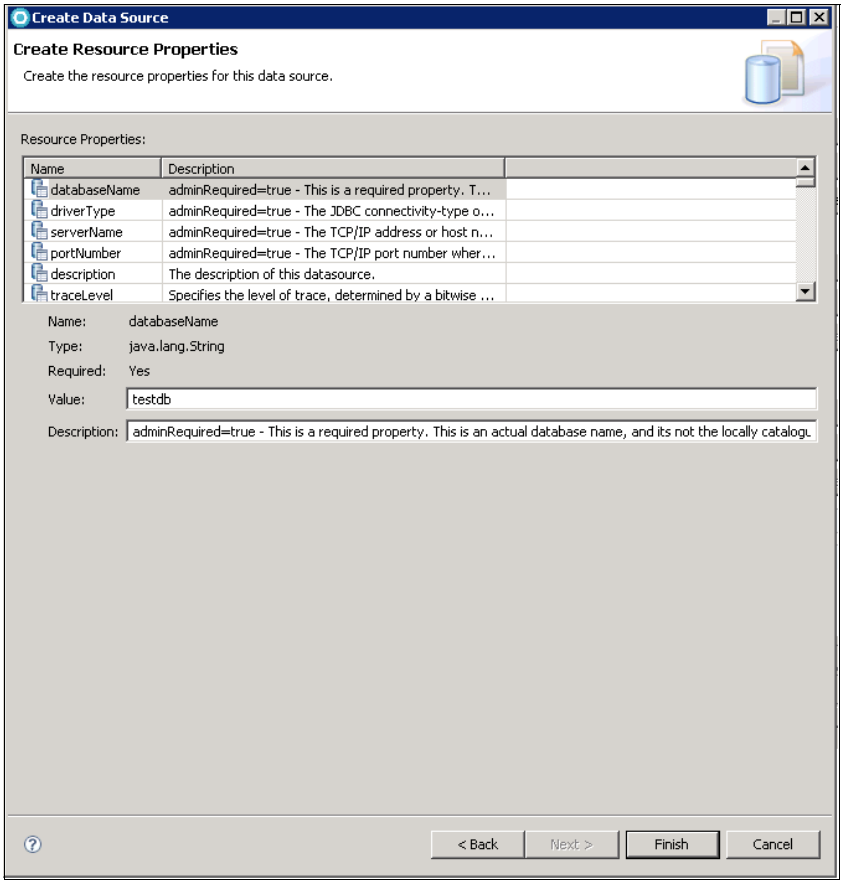


Figure 7-30 Resource properties for data source

14. The Data Sources section of WebSphere Application Server Deployment window now looks like Figure 7-31. Click CTRL+S to save changes.

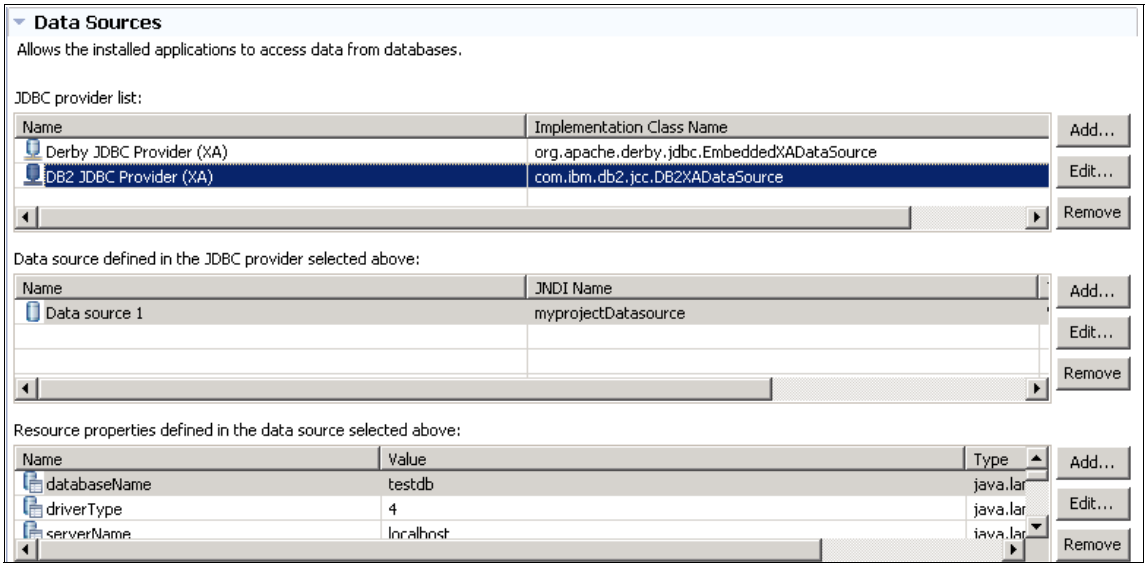


Figure 7-31 Data Sources panel of WebSphere Application Server deployment

Checking the project using migration toolkit

To run the Migration Tool on our project, perform the following steps:

1. Click the myproject application module. Right-click and select **Software Analyzer** → **Software Analyzer Configurations**. The Software Analyzer Configurations window is displayed. On the left panel, right-click **Software Analyzer** → **New**.
2. Click the **Rules** tab. Select **JBoss Application Migration** in the drop-down box.
3. Click **Set**. The respective rules become selected in the Analysis Rules and Domains area.
4. Click **Analyze**. After the analysis is complete, the Software Analyzer Results view is displayed, docked to the bottom panel. Inside the Software Analyzer Results view, you can see a tab for each domain of analysis. In our case, there are tabs named as follows: Java Code Review, XML File Review, and Class-path Review.
5. Expand the XML file review section and the results in this section. Remove the `jboss-app.xml` file by clicking `myprojecy\META-INF\jboss-app.xml` and clicking **Delete**.

6. Expand the class-path review section and the results in this section.Right-click one of the result lines and click **Quick fix all**.
7. Run the Software Analyzer again to ensure that no more warnings are generated.

Building and running the application on an integrated test environment

Migration is complete and the application can be run on WebSphere Application Server V7. We can check how the migrated application works using an integrated WebSphere Application Server V7 test environment.

To build/deploy and run the application, perform the following steps:

1. Click the `myproject_WEB\WebContent\index.html` file. Right-click and select **Run As** → **Run on server**. The Run on server dialog box is displayed.
2. Keep the default settings, and press **Finish**.

   After the application is deployed to test environment, an embedded web browser is displayed in the editor pane, presenting entry page of our seam-gen application.
3. Test the application by creating, updating, deleting, and listing entities. Use `admin` for the username with an empty password to log in.

7.4.7 Summary

While migrating our sample seam-gen generated application, we were able to demonstrate following concepts:

- Source migration to Rational Application Developer for WebSphere Software as a build and test environment.
- Use of WebSphere Application Server Migration Tool v1.1 for Rational Application Developer. Note that JBoss migration support in this tool is limited at the time of writing, but increasing rapidly.
- Using shared libraries for managing dependencies.

The main points to highlight during this migration are as follows:

- Differences in default JNDI bindings require rework on JNDI lookups. Seam frameworks requires a fair amount of configuration to work with WebSphere default JNDI bindings.
- Libraries that ship with JBoss Application Server distributions and not with WebSphere Application server create additional runtime dependencies that shall be taken care of.

- Using Hibernate for persistence requires transaction strategy configuration specific to WebSphere.
- Using a third-party JPA provider requires changes classloading configuration.

7.5 Migrating the Online Brokerage application

Online Brokerage is the sample application from the Apache Geronimo project. This application is used to demonstrate migration from JBoss Application Server to Geronimo Application Server. Because of this it includes JBoss-specific features. The version we have chosen is the one that was used in Geronimo V1.0 documentation. We feel this version contains a formerly common setup among JBoss users, namely J2EE 1.4, Hibernate (using JBoss specific HAR files), and JBoss Application Server v4.

Download and find more information about this application at the following web page:

https://cwiki.apache.org/GMOxDOC10/jboss-to-geronimo-hibernate-migration.html#JBosstoGeronimo-HibernateMigration-sampleApp

We chose to migrate this application for the following reasons:

- Use of the J2EE 1.4 application, which uses JSP and Hibernate
- Use of JBoss-specific HAR archives for Hibernate
- Possibly, this configuration still has a significant deploy base among JBoss users.

7.5.1 Migration approach

The Online Brokerage application is built using Apache Ant. In this particular migration exercise, we migrate the build environment to IBM Rational Application Developer for WebSphere Software, and use the build cycle that ships with this product. The migration is performed in the following steps:

1. Build an EAR package of applications for JBoss Application Server. Eliminate Geronimo dependencies in the build process to build the application without installing Geronimo Application Server.
2. Create a database for the application using DB2 and populate the database with data.
3. Import the EAR file that is built for JBoss to IBM Rational Application Developer for WebSphere Software, to create the project structure. Import source files into projects created from source distribution.

4. Analyze and fix problems reported by the build environment.
5. Build the Online Brokerage sample application using IBM Rational Application Developer for WebSphere Software. Deploy and test the application on the integrated WebSphere Application Server V7 test environment.

7.5.2 Building the Online Brokerage application

To build the Online Brokerage application archive file, perform following steps:

1. Download source distribution of the application from the following web page:

   https://cwiki.apache.org/GMOxDOC10/jboss-to-geronimo-hibernate-migration.data/brokerage.zip

1. Extract the downloaded archive file to a suitable location. We choose `C:\brokerage`. We will refer to this location as <brokerage_home>.
2. Open <brokerage_home>/build.properties file in an editor. Replace the line `jboss.home=E:/jboss-4.0.3SP1/server/hibernate` with the line `jboss.home=<jbossV4_home>/server/default`

Important: We are reusing the JBoss version 4.2.3 installation that we used wile migrating JSFEJB3 sample application. If you have skipped that exercise, see “Installing JBoss Application Server Version 4.2.3” on page 203

Also, be sure to use forward slashes as path separators when editing build.properties

3. In build.properties file, replace the line `dependency.dir=E:/hibernate-3.1/lib` with the line `dependency.dir=<jbossV4_home>/server/default/lib`.

   Save your changes.

4. Open the `<brokerage_home>/build.xml` file in an editor. Add the following line to the javac element of the jboss-compile Ant target, among other class path elements:

   ```
   <classpath path="${jboss.home}/lib/servlet-api.jar"/>
   ```

   Save your changes.

5. Open a command prompt, change the directory to `<brokerage_home>` and type **`ant`**.
6. After the ant build completes, the Online Brokerage application archive files are generated at `<brokerage_home>/jboss-artifact/brokerage.ear`.

7.5.3 Creating and populating the database

Because we are using IBM DB2 database, we use a slightly modified version of database scripts that are included in sample distribution. We work with the dbuser database user that we created at 7.4.2, “Preparing database schema” on page 227. Perform the following steps to create and populate the database

1. Click **Start** and select **Programs** → **IBM DB2** → **DB2Copy1 (default)** → **Command Line Tools** → **Command Editor**.
2. Enter the text in Example 7-6 in the command area, which is based on the `<brokerage_home>/db/db.sql` file

Example 7-6 Command area code snippet

```
CREATE DATABASE BRKRGDB;

CONNECT TO BRKRGDB USER DBUSER USING passwd;

CREATE TABLE "DBUSER  "."USERS"  (
        "USERID" VARCHAR(255) NOT NULL ,
        "NAME" VARCHAR(255) ,
        "PASSWORD" VARCHAR(255) ,
        "ADDRESS" VARCHAR(255) ,
        "CASH" DOUBLE ) ;

ALTER TABLE "DBUSER  "."USERS"
   ADD PRIMARY KEY
      ("USERID");


CREATE TABLE "DBUSER  "."STOCKS"  (
        "ID" VARCHAR(255) NOT NULL ,
        "NAME" VARCHAR(255) ,
        "PRICE" DOUBLE ) ;

ALTER TABLE "DBUSER  "."STOCKS"
   ADD PRIMARY KEY
      ("ID");


CREATE TABLE "DBUSER  "."USERSTOCKS"  (
        "ID" VARCHAR(255) NOT NULL ,
        "USERID" VARCHAR(255) NOT NULL ,
        "NAME" VARCHAR(255) ,
        "PRICE" DOUBLE ,
```

```
        "QUANTITY" INTEGER );


ALTER TABLE "DBUSER  "."USERSTOCKS"
   ADD PRIMARY KEY
      ("ID",
       "USERID");


INSERT INTO Users VALUES('srini', '6 Inch',
 'cheemu','Madivala Bangalore',10000);

INSERT INTO Users VALUES('j2ee', 'Lee Whittaker',
 'password','Tampa Florida',100000);

INSERT INTO userstocks VALUES('00000001', 'j2ee','BTDA',1,10);



INSERT INTO stocks VALUES('00000001','BTDA',1.00);
INSERT INTO stocks VALUES('00000002','Hitech Software',2.00);
INSERT INTO stocks VALUES('00000003','Bill s Cold Storage',3.00);
INSERT INTO stocks VALUES('00000004','Colonel s Fried Chicken',300.00);
INSERT INTO stocks VALUES('00000005','Toyo Motors',30.00);
INSERT INTO stocks VALUES('00000006','Timbuctoo Airlines',53.00);
INSERT INTO stocks VALUES('00000007','Happy Undertakers',73.00);
INSERT INTO stocks VALUES('00000008','Wacko Brothers',54.00);
INSERT INTO stocks VALUES('00000009','Mental Studios',456.00);
INSERT INTO stocks VALUES('00000013','ASFG',54.89);
INSERT INTO stocks VALUES('00000023','BFG',564.50);
INSERT INTO stocks VALUES('00000033','Mack',3444.50);
INSERT INTO stocks VALUES('00000043','Ronan & Sons',7.50);
INSERT INTO stocks VALUES('00000053','Bulls & Bears',553.40);
INSERT INTO stocks VALUES('00000343','HGHU',456.00);
INSERT INTO stocks VALUES('00000603','TTTM',77.00);
INSERT INTO stocks VALUES('00000463','GRASF',88.00);

COMMIT WORK;

CONNECT RESET;

TERMINATE;
```

3. Click **Execute**. A progress indicator is displayed. After the operations are completed, the results view displays the output, as in Figure 7-32.

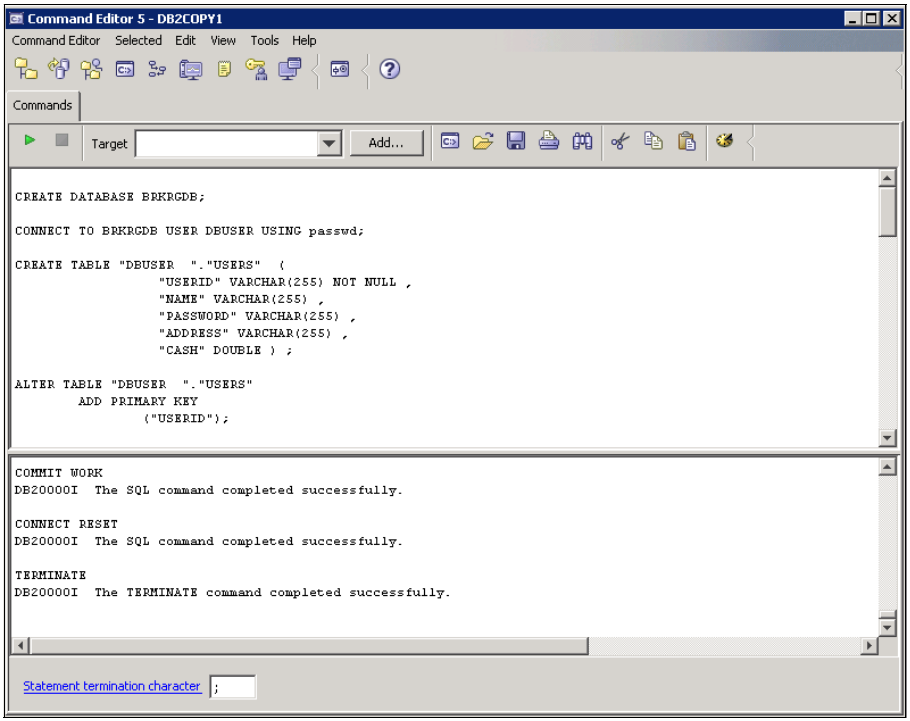


Figure 7-32 Creating brkrgdb database for sample application

7.5.4 Importing the EAR file and source code to RAD

There are several alternatives on how to import components in the HAR files into our project:

- The first alternative is to reuse generated HAR file in its binary form. Choose this alternative if you prefer to use other means of generating HAR files rather than Rational Application Developer for WebSphere Software.
- The second alternative is to create a utility module that contains contents of the HAR file, and on the web module add a dependency to this project. This alternative migrates all source code to Rational Application Developer for WebSphere Software as a second alternative, at the same time keeping persistence-related classes at a separate place. This serves the main motivation of using a HAR file in the first place.

- The third alternative is to import the contents of the HAR file (in our case the com.dev.trade.bo package and `hbm.xml` files) into a web module with other source code. This results in a simpler project structure, as the web module contains all source code.

In the procedures that follow, we demonstrate the second alternative. Follow the steps to do so:

1. Launch Rational Application Developer for WebSphere Software using **Start → IBM Software Delivery Platform → IBM Rational Application Developer 7.5 → IBM Rational Application developer**.
2. Launch the Import wizard using menu **File → Import.** Expand the JavaEE node in the import source tree and select the **EAR** file.
3. In the Enterprise Application Import window, browse to the location of the EAR file at `<brokerage_home>/jboss-artifact/brokerage.ear`. Click **Finish**.
4. Upon inspection we might see that brokerage_WEB module has imported classes under `Java Resources/Libraries/Web App Libraries/ImportedClasses`. We shall import the source files of these classes to the `Java Resources/src` folder in the same module and remove the class files from the project. Figure 7-33 shows the module structure before and after source code importation.

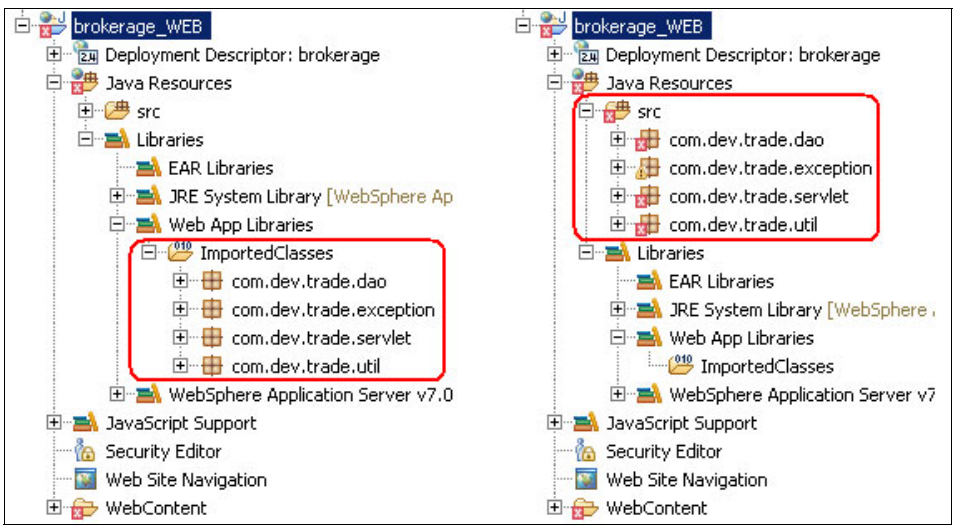


Figure 7-33 brokerage_WEB module before and after source code import

5. Click the **Java Resources/src** node in the myproject_EJB module, and select **File → Import** from the main menu.

6. In the Import window, select **General** → **File System.** The File system import window is displayed. Browse to the `C:\brokerage\src` folder and import the source code as seen on Figure 7-34.

Note: We are not importing com.dev.trade.bo package source code, as this package is packaged into the `brokerage.har` file, and is not a part of the web module.

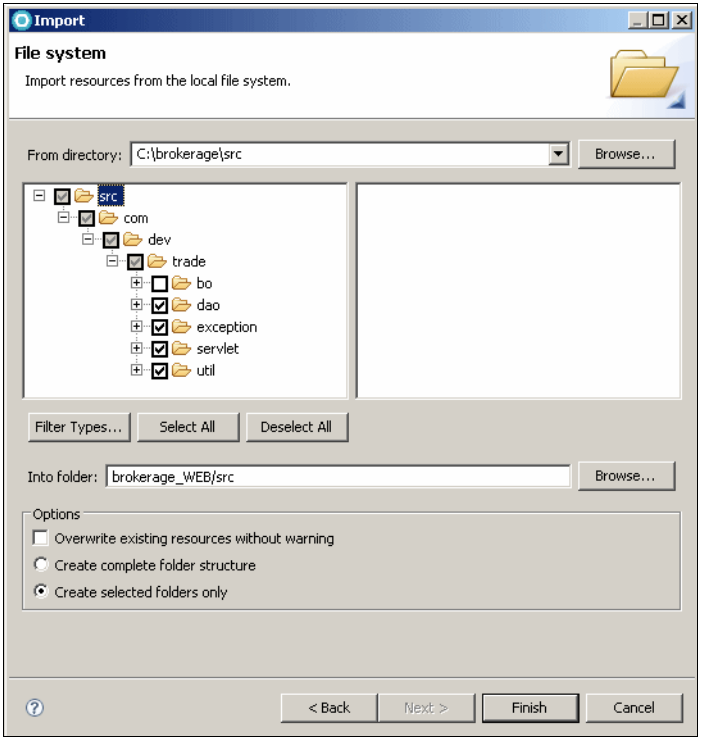


Figure 7-34 Importing source code files to brokerage_WEB module

7. Delete the packages under the `Java Resources/Libraries/Web App Libraries/ ImportedClasses` node by selecting them hitting **Delete**, and confirming the deletion. The project structure look like the right side of Figure 7-33 on page 259.

7.5.5 Analyzing and fixing the problems

We performed the following procedures to analyze and fix the migration problems

Fixing compile time and runtime dependencies

As can be seen on right hand side of Figure 7-33 on page 259, there are compile time errors (flagged with red cross icons) on several classes. These are due to missing third-party libraries such as Hibernate. Also, we are missing the dependency to the resources that are packaged if the HAR archive.

Perform the following steps to fix compile time and runtime dependencies:

1. Download Hibernate distribution from the following web page:

   `http://sourceforge.net/projects/hibernate/files/`

   At the time of writing, the latest version was 3.5.2-Final. Extract the file to a suitable location. We refer to this location as `<hibernate_home>`.

2. Download SLF4J distribution from the following web page:

   `http://www.slf4j.org/download.html`

   At the time of writing, the latest version was 1.6.0. Extract the file to a suitable location. We refer to this location as `<slf4j_home>`.

3. Select the `brokerage_WEB/ WebContent/ WEB_INF/ lib` folder. Right-click the selection to open the context menu and select **Import**. The Import wizard is displayed.

4. Select **General** → **File System** as the source and click **Next**. The File System window is displayed.

5. Click **Browse** and navigate to the folder containing the library and select the library to import. Do this for each library in Table 7-4.

Table 7-4 Required libraries

Location	Library
<hibernate_home>	hibernate3.jar
<hibernate_home/lib/required	antlr-2.7.6.jar
<hibernate_home/lib/required	dom4j-1.6.1.jar
<hibernate_home/lib/required	javassist-3.9.0.GA.jar
<slf4j_home>	slf4j-api-1.6.0.jar
<slf4j_home>	sl4j-simple-1.6.0.jar

6. To establish dependencies to resources in the HAR archive, rename the `brokerage.har` file in the brokerage application module root directory to `brokerage.jar`, by right-clicking this node and selecting **Rename**.
7. Open the brokerage_WEB module properties by typing ALT+Enter on this node.
8. Select **Java EE Module Dependencies**, and switch to the Web Libraries tab on this view.
9. Click **Add Jars**. The Jar Selection window is displayed.
10. Select `brokerage/brokerage.jar` and click **OK**. See Figure 7-35.

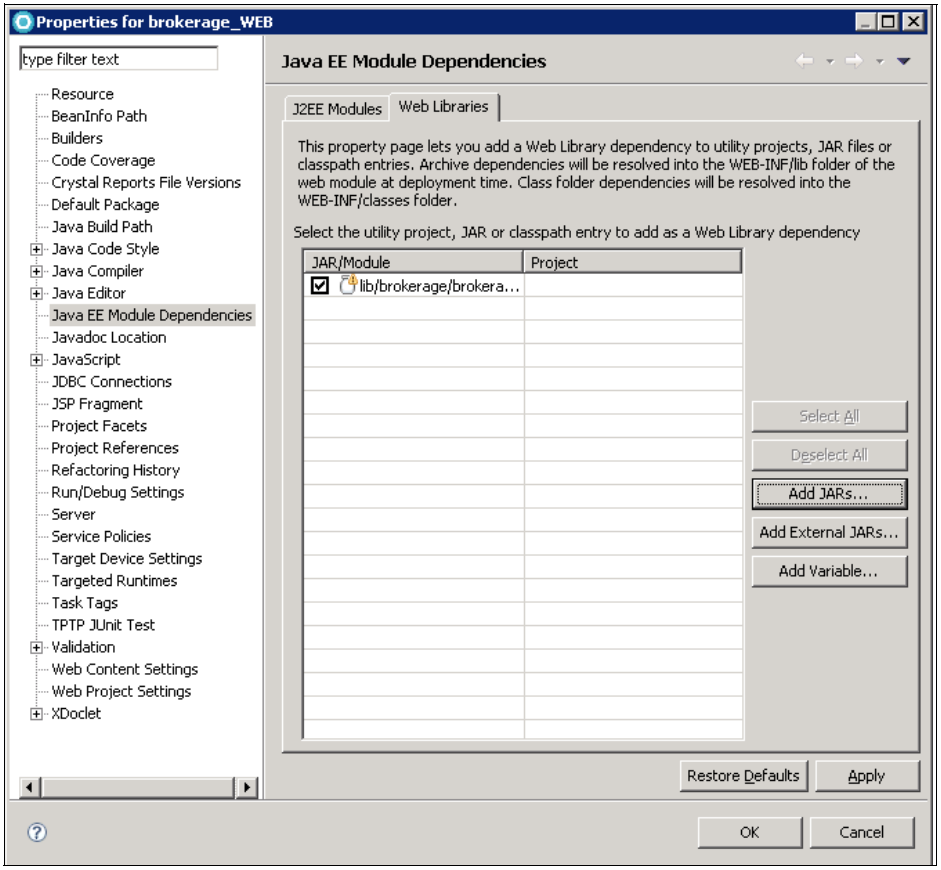


Figure 7-35 Adding dependency to HAR resources

11. Click **OK**. There are no more compile time problems.

Fixing Hibernate configuration

Tip: JBoss application server automatically processes HAR archives and configures the session factories, effectively binding them to the JNDI tree for further use by other parts of the application. Because this does not happen automatically on WebSphere Application Server, we must configure the session factories ourselves.

The first step in this process is to come up with a hibernate configuration file (`hibernate.cfg.xml`). A file with a similar purpose is already present in the META-INF directory of the HAR file (`hibernate-service.xml`). By using information in this file and the list of `hbm.xml` files in the HAR file, we can come up with a `hibernate.cfg.xml` file.

Because we might have more than one HAR archives in a project, it might be a good idea to name the `cfg.xml` file after the HAR file name as in our example.

1. Create a new file in the `brokerage_WEB/Java Resources/src` folder by right-clicking this node and select **New** → **Other**.
2. Select **XML** → **XML** and click **Next**.
3. On the filename field enter `brokerage.cfg.xml` and click **Finish**.
4. Add the code snippet in Example 7-7 as contents of this file and save your changes.

Example 7-7 Changes to the file

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<!--  name attribute is used for the JNDI name to be used for lookups
-->
<hibernate-configuration>
   <session-factory name="java:hibernate/BrokerageSessionFactory">

      <!--  Datasource properties, tailored for DB2 -.->
      <property
name="connection.datasource">java:comp/env/jdbc/HibernateDB</property>
      <property
name="hibernate.dialect">org.hibernate.dialect.DB2Dialect</property>
      <property name="hibernate.default_schema">dbuser</property>
```

```
      <!--  WebSphere specific properties -->
      <property
name="hibernate.transaction.manager_lookup_class">org.hibernate.transac
tion.WebSphereExtendedJTATransactionLookup</property>
      <property
name="hibernate.transaction.factory_class">org.hibernate.transaction.JT
ATransactionFactory</property>
      <property name="jta.UserTransaction">
java:comp/UserTransaction</property>

      <!-- Mapping files (for each hbm.xml file in HAR file) -->
      <mapping resource="Stock.hbm.xml" />
      <mapping resource="UserStock.hbm.xml" />
      <mapping resource="User.hbm.xml" />
   </session-factory>
</hibernate-configuration>
```

The following points relating to Example 7-7 on page 263 require your attention:

- Notice that name attribute of the session-factory element is used later for looking up session factory.
- Because we are using IBM DB2 as our test database, we add respective dialect.
- We add WebSphere specific properties related to transaction strategy configuration according to article at the following web page:

  http://www.ibm.com/developerworks/websphere/techjournal/0609_alcott/0609_alcott.html
- We add a mapping element for each hbm.xml file in our HAR archive.

5. Ensure that the Hibernate session factory defined in the `brokerage.cfg.xml` file is initialized properly before any part of the code looks up for Hibernate session factory using JNDI. The Online Brokerage application contains a single servlet that acts as a front controller, so a good place to initialize the session factory is the init method of this servlet.

Note: For larger applications, using servlets to initialize session factories might not be a feasible solution. In this case use startup beans. Startup beans allow business logic to run when an application starts or stops and gives the opportunity to initialize session factories. Details on how to use startup beans can be found at following web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/asyncbns/tasks/tasb_confstb.html

6. Open `TradeDispatcherServlet.java` in an editor by double clicking it at `brokerage_WEB/Java Resources/ src` at `com.dev.trade.servlet` package.
7. At the init method, after the `super.init();` line, add the following line:

   ```
   new Configuration()
   .configure("brokerage.cfg.xml").buildSessionFactory();
   ```

   The editor marks the Configuration class as not found.
8. Import the org.hibernate.cfg.Configuration class by clicking CTRL+1 on the configuration and selecting this class in the drop-down list.
9. Save the TradeDispatcherServlet by typing CTRL+S.

Creating the data source

As seen in `brokerage.cfg.xml`, our hibernate configuration depends upon the existence of a data source with the JNDI name `jdbc/HibernateDB`. Perform the following steps to create this data source.

1. Select the brokerage application module and right-click for the context menu. Select **Java EE** → **Open WebSphere Application Server Deployment**.
2. Scroll down to the Authentication section. Click **Add** near the JAAS Authentication list subsection. The Add JAAS Authentication Entry dialog box is displayed.
3. Enter `db2user` for alias, user ID, and description fields, and `passwd` for password field. Click **OK**.

   At this point new authentication alias is listed in JAAS Authentication list.
4. Scroll up to Data Sources section, click **Add** near the JDBC provider list subsection. The Create JDBC Provider window is displayed.
5. Select `IBM DB2` as the database type and `DB2 Universal JDBC Driver Provider (XA)` as the JDBC provider type. Click **Next**. The JDBC Provider details window is displayed.
6. Enter `DB2 JDBC Provider (XA)` in the name field.

7. Select the entries in the classpath section and click **Remove**.
8. Click **Add External JARs**. A file selection dialog box is displayed. Navigate to the `C:\Program Files\IBM\SQLLIB\java` folder and select the following JARs:
   - `db2jcc.jar`
   - `db2jcc_license_cu.jar`

   Click **OK** to add.

   The JDBC provider settings window looks like Figure 7-36.

   Click **Finish**.

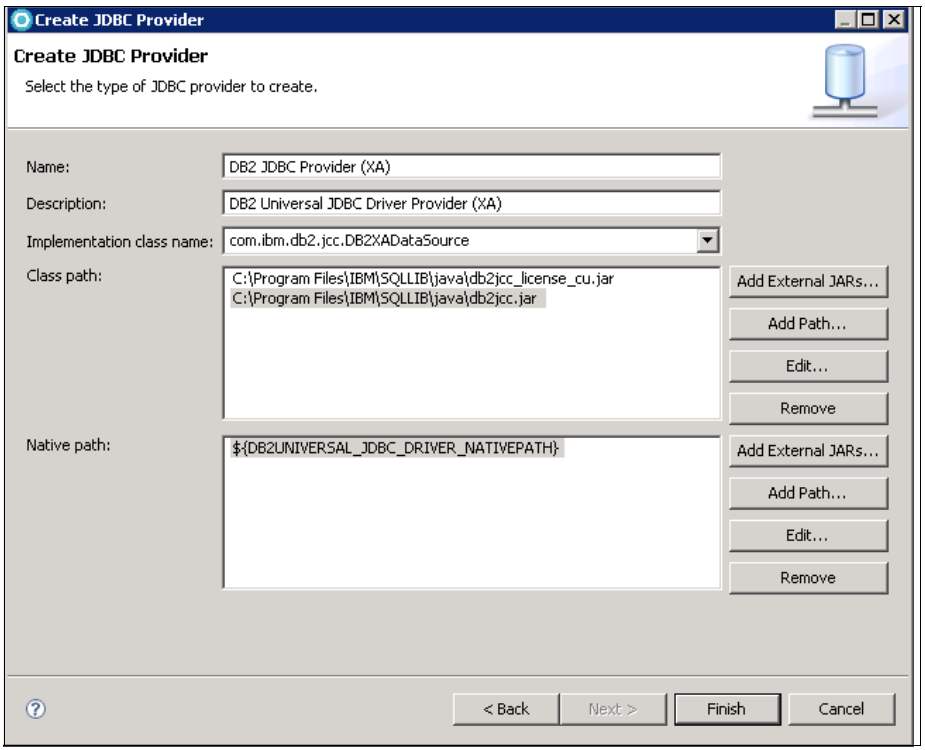


Figure 7-36 JDBC provider settings

   At this point the new JDBC provider is listed in JDBC providers list.

9. Click this line to select `DB2 JDBC Provider (XA)`, and then click **Add** in the data sources subsection. The Create Data Source window is displayed.
10. Select `DB2 Universal JDBC Driver Provider (XA)`. Click **Next**. The Data Source details window is displayed.
11. In the JNDI Name field, type `jdbc/HibernateDB`.

12. In the Container-managed authentication alias and Component-managed authentication alias fields, select the dbuser authentication alias that we have created before using combo box. The window looks like Figure 7-37. Click **Next**.

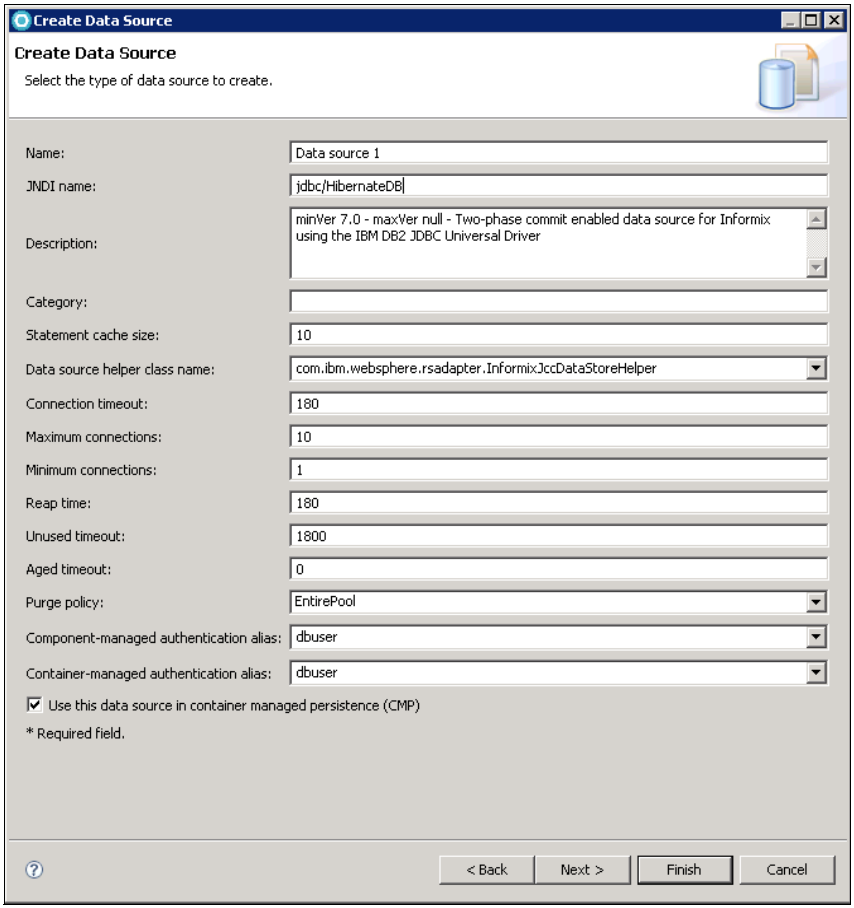


Figure 7-37 Data Source details

13. Resource properties window is displayed. Two values we shall define are `databaseName` and `serverName`. Specify `brkrgdb` for `databaseName`. Similarly, define `localhost` for `serverName`. Click **Finish**.

    The Data Sources section of WebSphere Application Server Deployment window now looks like Figure 7-38. Click CTRL+S to save your changes.

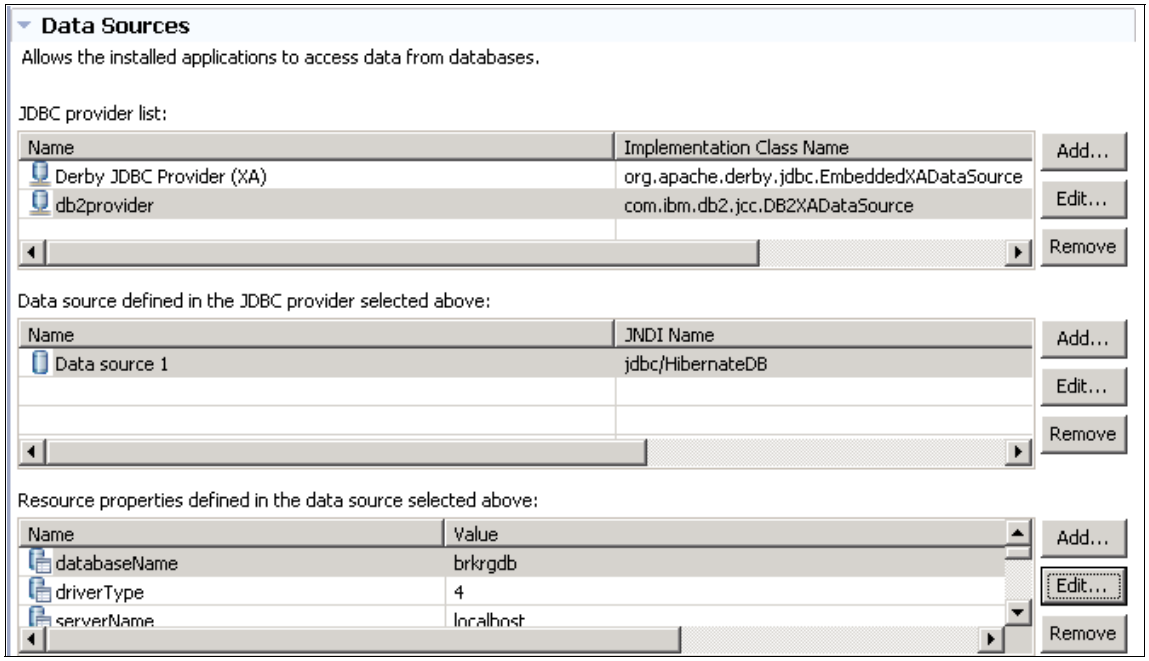


Figure 7-38 Data Sources panel of WebSphere Application Server Deployment

14. Attach the data source to the already defined resource reference in the web module. Open the brokerage_WEB module deployment descriptor in the editor by double clicking **brokerage_WEB/Deployment Descriptor: brokerage**.
15. Switch to the References tab and click **Add**. The Reference window is displayed.
16. Select **Resource Reference** and click **Next**.
17. On the References view, click **ResourceRef jdbc/HibernateDB**.

18. In the WebSphere Bindings section, enter `jdbc/HibernateDB` in the JNDI name field. See Figure 7-39.

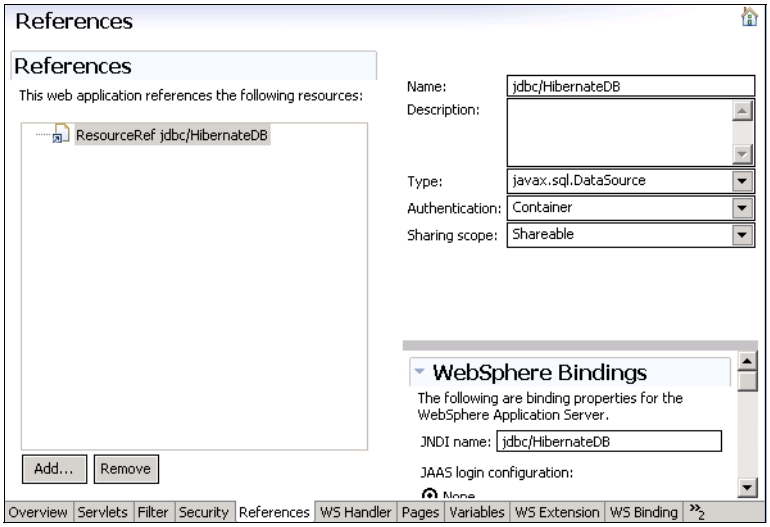


Figure 7-39 Resource references for brokerage_WEB

19. Save your changes by typing CTRL+S.

Building and running the application on an integrated test environment

At this point, migration is complete and the application can be run on WebSphere Application Server V7. We can check how the migrated application works using the integrated WebSphere Application Server V7 test environment.

To build, deploy, and run the application, perform the following steps:

1. Click the `brokerage_WEB\WebContent\login.html` file. Right-click and select **Run As** → **Run on server**. The Run on server dialog box is displayed. Keep the default settings. Press **Finish**.

   After the application is deployed to test environment, an embedded web browser is displayed in the editor pane, presenting entry page of the application.

2. Test the application by registering a new user for yourself and buying and selling stock.

Importing HAR contents in source form

As mentioned in 7.5.4, “Importing the EAR file and source code to RAD” on page 258, the configuration at this point reuses the HAR archive in its binary form. We might also import HAR contents in source form into Rational Application Developer for WebSphere Software. To so, we add a new project with the contents of the HAR file, and configure a dependency to this new project at the web module. This way, developers can take full advantage of Rational Application Developer for WebSphere Software, and still keep persistence logic as a coherent unit, which is probably the intention for packaging this unit into a HAR archive in the first place.

1. Create a new Utility project. Right-click the brokerage application module and select **New** → **Other**. In the Wizard selection window, select **Java EE** → **Utility Project**.
2. Enter `brokerage_persistence` as the module name. See Figure 7-40. Click **Finish**.

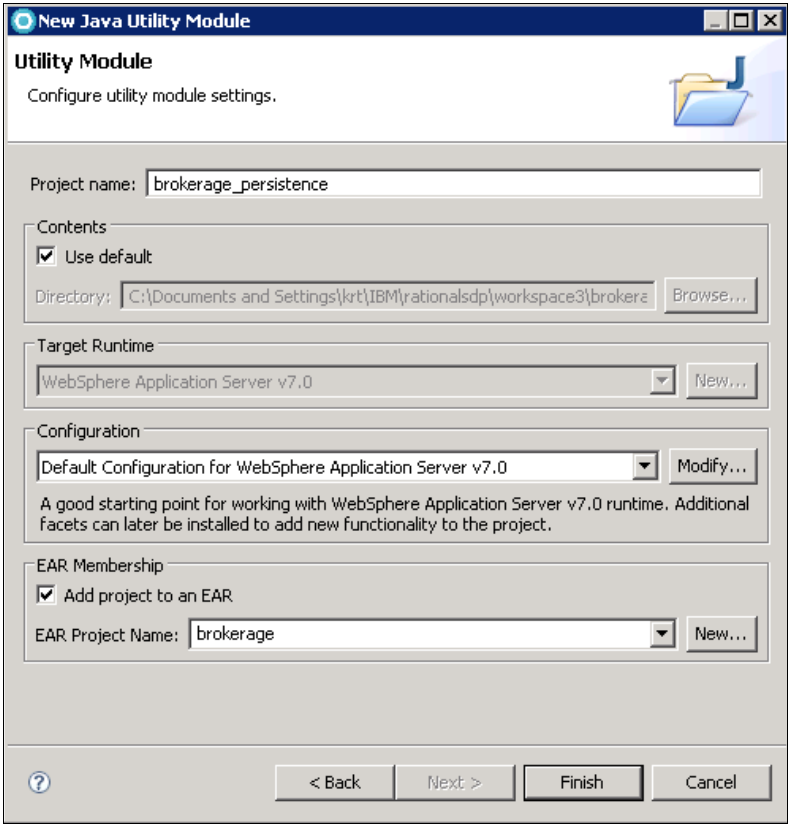


Figure 7-40 Creating a new utility module

3. Import com.dev.trade.bo package from <brokerage_home>/src into the brokerage_persistence/src folder using the import wizard. See Figure 7-41.

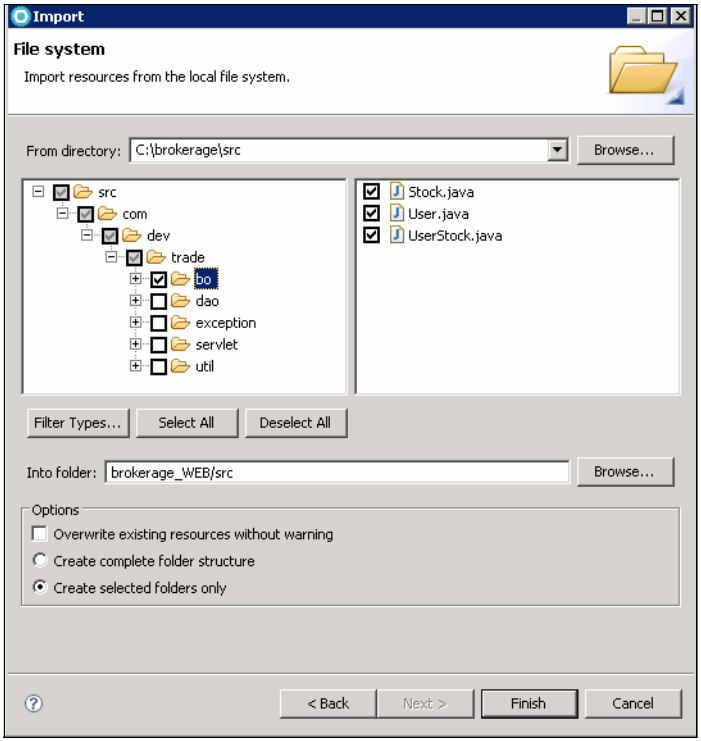


Figure 7-41 Importing source files to utility project

4. Import hbm.xml files from the <brokerage_home>/hibernate folder into the brokerage_persistence/src folder using the import wizard. At this point, the utility module look like Figure 7-42.

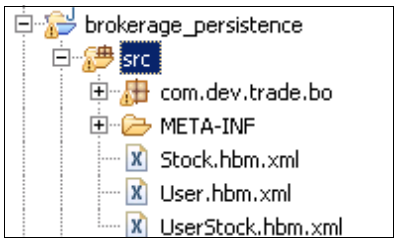


Figure 7-42 Utility module contents

5. Open the brokerage_WEB module properties by right-clicking and selecting **Properties**.

6. In the Properties window, select **Java EE Module Dependencies**. Switch the Web Libraries tab. Clear the dependency to brokerage.jar and select the dependency to brokerage_persistence module. See Figure 7-43. Click **OK**.

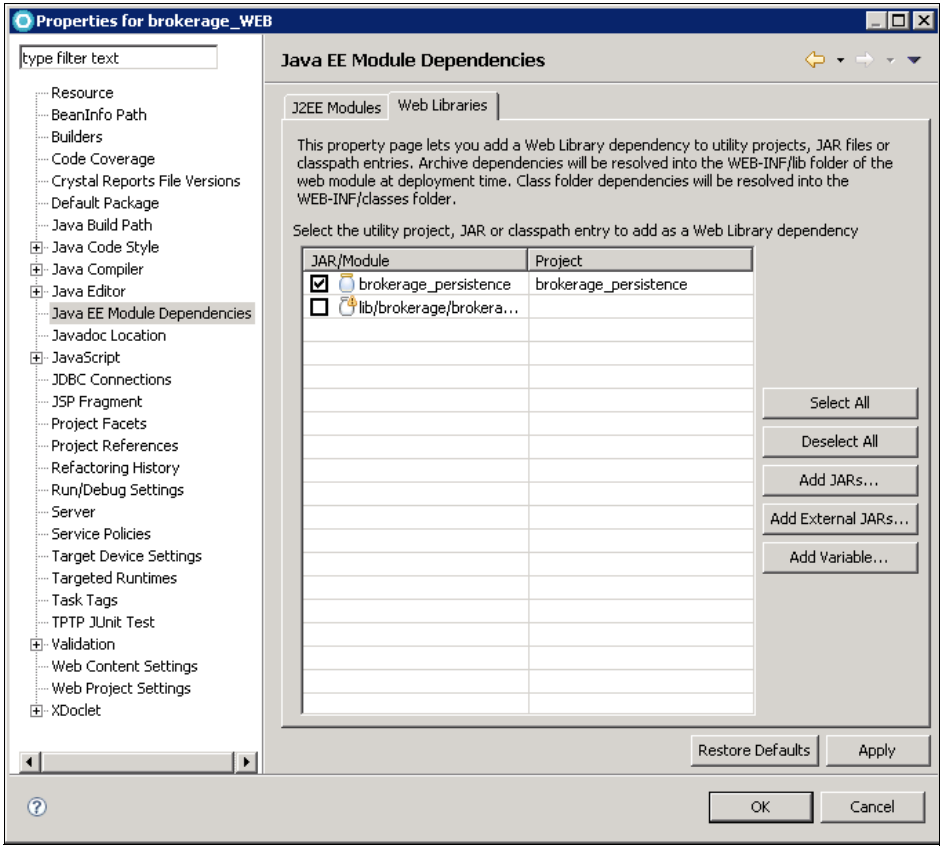


Figure 7-43 Module dependencies of web module

7. Delete `persistence.jar` from the persistence application module.
8. The migration is complete. Test the application as described at “Building and running the application on an integrated test environment” on page 253.

Summary

While migrating our sample seam-gen generated application, we were able to demonstrate following concepts:

- Source migration to Rational Application Developer for WebSphere Software as build and test environment.
- Several alternatives on organizing project structure for Hibernate applications using HAR archives.
- Configuration of Hibernate for WebSphere.

The main points to highlight during this migration are as follows:

- Using Hibernate for persistence requires transaction strategy configuration specific to WebSphere
- Initializing session factories is not automatic on WebSphere. We must explicitly initialize session factories by startup beans or other means.

8

Migrating from Apache Tomcat

In this chapter, we describe how to migrate two J2EE web applications from Apache Tomcat 6.0.26 to WebSphere Application Server V7. There are several challenges associated with such migrations. One of them is the fact that Tomcat applications are often built on top of open source frameworks and libraries.

Naturally, there is a concern that those frameworks might not work on commercial applications servers. The other challenge is that the development and build environment is often command line-based and uses Ant or Maven tools with no dependency on any particular IDE. When we started this book, we ran a search on the web for open source Java web applications. We have selected two typical Tomcat applications that we found on sourceforge.net to reflect real world experience during migration.

This chapter has the following sections:

- "Introduction" on page 276
- "Prerequisites and assumptions" on page 276
- "Software installation" on page 277
- "MvnForum migration" on page 279
- "Easy JSP Forum migration" on page 289

© Copyright IBM Corp. 2010. All rights reserved.

8.1 Introduction

Apache Tomcat 6.0.26 is an open source implementation of Sun's J2EE web container. It implements Java Servlet and JSP specifications and is used in the official reference implementation for the Java Servlet and JavaServer Pages specifications. Tomcat is developed by a community of developers under the umbrella of the Apache Software Foundation (ASF). Tomcat is widely used by a large number of web sites. It is popular for its small footprint, simplicity, and the large number of toolkits and frameworks that run on it. One of its nice features is that it is open source and is free to use.

Applications tend to grow over time and need higher quality of services (QoS) and more sophistication than Tomcat can offer (such as centralized administration for clustered domains, failover, advanced role-based security for users and administrators, script-based administration, advanced monitoring tools, portal infrastructure, page fragment and distributed Java caching, transactions, and so forth). That is why applications need to be redeployed or migrated into enterprise-grade application servers, such as WebSphere Application Server V7.

The two applications we have selected for migration are built on top of a number open source frameworks:

- mvnforum-1.2.1-mvnad-1.0.1
- Easy JSP Forum 1.0

8.2 Prerequisites and assumptions

There are prerequisites and assumptions we need to consider before beginning with the migration examples. One basic requirement is that the reader should have the destination environment installed and configured as described in Chapter 5, "Installation and configuration" on page 87. This provides initial background knowledge if the reader is not yet familiar with IBM products

The reader should also be familiar with the J2EE specification and architecture, and be able to understand and run basic SQL commands. The reader should be familiar with WebSphere Application Server Toolkit V6.1. The reader must also be knowledgeable about Apache Tomcat 6.0.26.

The following software should be installed before starting the migration:

- Java SE 6

  We use Java SE 6 because it is supported by WebSphere Application Server V7 and we need to make sure our applications work well in this Java virtual machine (JVM). The WebSphere Application Server V7 installation includes Java Development Kit (JDK) 1.6.0, so you do not need to install additional JDKs on your machine.

- Apache Tomcat 6.0.26

  This version was chosen because it was the latest version available at the time of writing this IBM Redbooks publication. The Tomcat home page advises all Tomcat users to upgrade to Tomcat 6.x whenever possible.

- IBM WebSphere Application Server V7

  Before migrating to WebSphere Application Server V7, familiarize yourself with the sample applications shipped with the product. Also, examine the publications listed in “Related publications” on page 377.

- IBM UDB DB2 9.5

  WebSphere Application Server V7 supports many databases, but we have decided to use DB2 for MvnForum. Instructions on how to install and configure DB2 Universal Database V9.5 can be found in Chapter 5, “Installation and configuration” on page 87.

- WebSphere Application Server Toolkit V6.1

  WebSphere Application Server Toolkit V6.1 was only used for the Easy JSP Forum. The official distribution available at sourceforge.net is not an enterprise package application model. It is required to import to WebSphere Application Server Toolkit V6.1 to create the EAR file.

8.3 Software installation

This section provides high-level steps and tasks for installing and configuring Tomcat, Maven, Ant, and other products to run our sample applications in their original environment.

8.3.1 Apache Tomcat

Detailed instructions for installing, configuring, and managing Apache Tomcat 6.0.26 are provided in the product documentation guides available at the following web page:

http://tomcat.apache.org/tomcat-6.0-doc/index.html

The following steps illustrate the process we following to install and configure our initial environment as a starting point for deploying sample applications.

1. Download Apache Tomcat 6.0.26 binary distribution for Windows from the following web page:

   http://tomcat.apache.org/download-60.cgi

   The file downloaded was `apache-tomcat-6.0.26-windows-x86.zip` (6.65 MB)

2. Extract all archives into the same directory (Referred to as `<tomcat_home>`):

   `C:\apache-tomcat-6.0.26`

3. Create an environment variable pointing to the Java included in WebSphere Application Server V7. Go to the Windows Desktop and navigate to **My Computer** → **Advanced Tab** → **Environment Variables** → **System Variables** and click **New.** In the variable name enter `JAVAHOME` and a value of `C:\WebSphere7\AppServer\java.`

   This helps us to ensure that our sample application runs on the same JDK in both environments and might eliminate potential JDK compatibility issues.

4. Install the DB2 JDBC driver. The DB2 driver is used by our applications for connecting to DB2 Universal Database V9.5. Copy `db2jcc.jar` and `db2jcc_license_cu.jar` from the `<db2_home>java` directory to the `<tomcat_home>\lib` directory.

5. Go to your `<tomcat_home>\bin` directory and run the **`startup.bat`** command to start the Apache Tomcat 6.0.26 server.

6. Once the server is started, verify that it is running by opening a web browser and pointing it to this URL: `http://localhost:8080`. You should see the Tomcat Welcome window.

8.4 MvnForum migration

The MvnForum open source edition is used in this book. This application implements the bulletin board, also known as the forum. The product was developed to run in most of J2EE servers and databases. In this book we used the DB2 as the back-end database. The MvnForum application works on UNIX, Linux, and Windows and is implemented as a J2EE 1.4 web application. This application is distributed with the full source code under the terms of the GNU General Public license. It is an open source, powerful, easy to use, easy to set up bulletin board (forum) built on the Java J2EE technology (Jsp/Servlet). It means that you can use it free of charge to build your own discussion communities. MvnForum is compatible with any servlet containers that support Jsp 1.2 and Servlet 2.3.

We found this to be an interesting application to be migrated for the following reasons:

- It is based on popular open source frameworks.
- It was designed for running on Tomcat, JBoss, WebSphere Application Server, and other popular J2EE application servers.
- It has never been tested on WebSphere Application Server V7.
- The application has advanced functionality and is fairly large in size. It uses J2EE and J2SE functionality (such as JSP 1.2) with a number of custom tags, Servlet 2.3, JSTL 1.0.2, and so forth.

Overall, we believe this application is representative of a typical ISV build if they were using Tomcat or JBoss as their deployment platform.

8.4.1 Migration approach

The MvnForum application was built with non-IBM development tools using Ant as a build tool. The application will have to be enhanced in the future and it needs to support Tomcat and WebSphere as deployment targets. The decision has to be made about what tool to use to migrate the application and what tools are used to continue development of this application going forward.

For this particular migration project we decided to keep the existing development environment. The existing application was built by a team already familiar with the current build and development environment and has best practices built around it. It might be more productive to take advantage of the developers current skills and continue using existing processes, frameworks, and tools, assuming they work well for both environments (Tomcat and WebSphere).

The distribution package follows the WAR directory structure. Use a compression tool to create the WAR file. You can go to the distribution directory and select mvnplugin, WEB-INF directories, `index_mvnforum.jsp`, and `index.jsp` and create a compressed file with this content. After that, rename it with a WAR extension.

We want to show that it is possible to migrate an application, such as MvnForum, to WebSphere Application Server V7 without using an IDE. We also want to demonstrate that it is possible to maintain the original development environment.

8.4.2 Configuring the source environment

Make sure that the application works properly in the source environment before migrating it to WebSphere. This section describes how we configured Apache Tomcat 6.0.26 and MvnForum to run in the source environment. Instructions on how we installed the Apache Tomcat 6.0.26 environment are described in 8.3, “Software installation” on page 277.

Getting the MvnForum application

Before starting with the migration, obtain source files for the MvnForum application. Download the `mvnforum-1.2.1-mvnad-1.0.1-bin-20081120.zip` file with all source files from the following web page:

`http://sourceforge.net/projects/mvnforum/files/`

Create a source home directory and extract the application source code you just downloaded. From now on, we refer to this directory as `<source_home>`.

Creating the DB2 database

The MvnForum 1.21. is compatible with nine database managers including DB2. The tables script creation is provided. Perform the following steps to create the DB2 database:

1. Create an empty database with default options.
2. Get the `mvnForum_DB2.sql` file from the `sql_mvnforum` directory in the source distribution of MvnForum.
3. Create the database and the tables run the commands listed in Example 8-1 at the DB2 command window.

Example 8-1 Database creation commands

```
db2 create db MVNFORUM
db2 connect to MVNFORUM user db2admin using your_password
db2 -tvf mvnForum_DB2.sql
```

Deploying the application

Once you have installed DB2 and created the tables using the provided table creation script, deploy the application into Tomcat. Tomcat automatically deploys web applications located in the webapps directory by default. Follow these steps to deploy the MvnForum application:

1. In the application source, go to `WEB-INF\classes`. There are two xml files that fit to your environment:

   - `mvncore.xml`
   - `mvnforum.xml`.

   In our scenario we replaced the `mvncore.xml` file with the code listed in Example 8-2 to fit the database requirements.

Example 8-2 Changing mvncore.xml file code snippet

```
<driver_class_name>com.ibm.db2.jcc.DB2Driver</driver_class_name>
<database_url>jdbc:db2://IBM-6474L1BB23C:50000/MVNFORUM</database_url>
<database_user>fxavier</database_user>
<database_password>password_here</database_password>
```

2. Create the WAR file and copy it to the `<tomcat_home>\webapps` directory.
3. From the command line, change to the Tomcat directory: `<tomcat_home>\bin`.
4. If you have already deployed MvnForum into Tomcat, stop Tomcat by running the **`shutdown.bat`** command.
5. Remove the previously deployed application from the `<tomcat_home>\webapps` directory. Make sure to remove both the `mvnforum.war` file and the mvnforum directory (exploded war file).
6. Start Tomcat by running the **`startup.bat`** command:

   You should see no errors in the log if the application was deployed successfully.
7. Make sure that you have JavaScript enabled in your web browser.

   Tomcat builds a root URI for the web application based on the name of the war file. Therefore, you can access the application at the following URL (assuming you kept default port for the Tomcat installation and did not rename the war file):

   `http://localhost:8080/mvnforum`

   To log in to the application, use username `admin` and password `admin`.

Testing the application

The script that creates the database tables required for this application is also downloaded with the application.

We tested the application manually, to verify all the functionality of the application. We logged in to the application, and changed the profile avatar picture and email address.

After that simple test, we checked the log files and did not find any errors. This simple test was implemented to ensure that the application was up and running without configuration problems. This is obviously not recommended for a production environment.

At this point, we have successfully built, deployed, and tested MvnForum running on Apache Tomcat 6.0.26, which is its original environment. Having the application working properly on the source environment is our starting point for migrating it to WebSphere Application Server V7.

8.4.3 Migrating the MvnForum application

This section describes the steps necessary for migrating the application from the source environment to the WebSphere Application Server V7 environment. We use the same database created in Tomcat environment.

The migration steps for this applications do not include source code changes, just the `mvncore.xml` file to adjust the WebSphere data source, server address, and webcontainer port number. We need to configure the WebSphere Application Server resources to install the application and, after they are configured, deploy and test the application.

Changing the default connection pool to WebSphere data source

When we performed the migration, we moved in small steps. We changed the minimum number of properties before moving forward to the next step. We tried to make sure that the baseline worked well (the application runs in Tomcat) and then we tried to deploy the same configuration to WebSphere Application Server V7.

This is a useful technique to isolate changes and have a more controlled environment. However, in this book, we are not taking you through all the steps in the exact order we used (we simplified the process and omitted the trial and error). Instead, we show you how to make all changes first and then build and

deploy the application in one step. Usually, this is an iterative process, but to simplify the description, we are not showing all intermediate deployments and builds that we performed.

Considering that the application works with DB2 using the default connection pool, you need to change the connection pool mechanism. The default connection pool provided in MvnForum is not designed for production use, so you must use the WebSphere-provided connection pool.

There are many benefits to using the WebSphere connection pool implementation. The connections are obtained through JNDI from the WebSphere data source and automatically participate in the container-managed transactions of the application server.

To implement these changes, we need to update the `mvncore.xml` configuration file. To use WebSphere Application Server V7 data source we need to change two xml parameters. The file is located in the following directory, and is described in Example 8-3.

```
<source_home>\WEB-INF\classes\mvncore.xml
```

Example 8-3 Define data source in mvncore.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <baseElement>
- <dboptions>
   ...
   <use_datasource>true</use_datasource>
   ...
   <datasource_name>jdbc/dsmvnforum</datasource_name>
   ...
</dboptions>
</baseElement>
```

Note: We have only two properties defined here. These are the only properties needed by MvnForum to get connected to the database.

When developing an application, you generally do not know about the name of the data source on the target application server. In your code, you do not look up the data source directly. Instead, you look up the resource reference from the `java:comp/env` namespace file. The JNDI resource jdbc/dsmvnforum is configured in next section.

We need also to modify context path and server path to fit the WebSphere webcontainer context and port number. In our case the webcontainer is used in port 9080 and the default context path is `/mvnforum/mvnforum`. The trusted domains entry should also be changed to fit your environment settings.

The trusted domains entry should also be changed to fit your environment settings as shown in Example 8-4.

Example 8-4 Define paths and trusted domains in mvncore.xml

```
<paramoptions>
  <context_path>/mvnforum/mvnforum</context_path>
  <server_path>http://localhost:9080</server_path>
  </paramoptions>
</baseElement>
...
<allow_http_referer_prefix_list>http://localhost:9080;http://127.0.0.1:
9080</allow_http_referer_prefix_list>
...
```

Packaging the application

The distribution application code does not require a build to install in WebSphere Application Server V7, you can get the code, select all directories and files from source home and create a compressed file then rename it to a war file. With all these modifications in place, you are now ready to deploy the MvnForum application so that you have a war file ready for deployment to WebSphere Application Server V7.

Creating a new data source

Follow these steps to configure MvnForum to use the WebSphere Application Server V7 data source. We assume that you have the server1 application server up and running.

Note: For each migration exercise, we created a new WebSphere profile so that we have a clean environment for deploying the migrated applications. The creation on WebSphere profiles is optional.

1. Open the WebSphere Administrative Console:

   ```
   http://localhost:9060/ibm/console
   ```

2. Log in and click **Resources** → **JDBC** → **JDBC Providers** and then **New** to create a new JDBC provider with the characteristics shown in Table 8-1.

Table 8-1 JDBC provider values

Parameter	Value
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	Connection pool data source
Name	DB2 Universal JDBC Driver Provider

3. Click **Next**. Specify the correct path to the DB2 JDBC driver and click **Apply**. In our migration example, the path to the JDBC driver is as follows:

   ```
   C:\Program Files\IBM\SQLLIB\java
   ```

4. Click **Finish**, then save the configurations to the master configuration.
5. Click **Resources** → **JDBC Data sources**. Select the scope and click **New**. Create a new data source with the characteristics listed in Table 8-2.

Table 8-2 Data Source values

Parameter	Value
Data source name	DSMNVFORUM
JNDI Name	jdbc/dsmvnforum

6. Click **Next**.
7. Select the **Select an existing JDBC provider** radio button and select **DB2 Universal JDBC Driver Provider**. Click **Next**.

8. In the “Enter database specific properties for the data source” panel, enter the values listed in Table 8-3.

Table 8-3 Database specific properties

Parameter	Value
Driver type	4
Database name	MVNFORUM
Server name	ibm-6474l1bb23c
Port number	50000
Use this data source in container managed persistence (CMP)	Checked

9. Click **Next**.
10. In the Setup security aliases panel, click in **Next**. In next window, review the summary and click **Finish**.
11. Return to the data source you created, Click **JAAS - J2C authentication data** in the Related Items panel.
12. Click **New**.
13. Create a new alias and specify the values listed in Table 8-4.

Table 8-4 J2C alias configuration

Parameter	Value
Alias	JAASMVNFORUM
UserId	db2admin
Password	your_password

14. Click **OK** and return to the data source you created. From the Component-managed authentication alias drop-down menu, select the J2C alias you created.
15. Verify the connection to the database. Select the data source you created and click **Test connection**.

Deploying the application

There are several ways to deploy applications and create resources (data sources, and so forth) in WebSphere Application Server V7. Usually, you use the WebSphere Administrative Console to perform these tasks for the first time, but for production environments, you would normally write a script with the wsadmin tool instead.

For a development environment, you can automatically deploy applications from WebSphere Application Server Toolkit V6.1 with a single click. Because we are not using the Rational product, we performed a manual deployment. If we had to deploy the application multiple times (as in the real world), we can use a wsadmin script and make it part of our build process. For more information about ways of deploying applications, visit the following web page:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/crun_app_install.html

In this case, we only have one application, and all the necessary resources are already configured in the WebSphere server1 instance, so we can deploy the application through the WebSphere Administrative Console, following these steps:

1. From the WebSphere Administrative Console, click **Applications** → **New Application** → **New Enterprise Application**.
2. Specify the path to the WAR file you just created and the context root as described in Table 8-5 and click **Next**.

Table 8-5 Installation values

Parameter	Value
Local File System Path	<source_home>\package\war\target\mvnforum.war
Context Root	/mvnforum

3. Select **Fast Path - Prompt only when additional information is required**.

   Expand **Choose to generate default bindings and mappings**, and select **Generate Default Bindings**. Click **Next**.
4. In the Step 1: Select installation options window, leave all the default options and click **Next**.
5. In the Step 2: Map modules to servers window, leave all the default options and click **Next**.
6. In the Step 3: Map context roots for web modules window, enter `/mvnforum` in the Context Root field, and click **Next**.

7. In the Step 4: Summary window, click **Finish**. It might take a minute to deploy the application and validate all mappings. After you are finished, save to the master configuration.

Next, start the application.

1. From the WebSphere Administrative Console, click **Applications** → **Application Types** → **Enterprise Applications**, and select the **MvnForum** application check box. Click **Start**.

   You should see the application started and a green arrow to the right of the application name, as shown in Figure 8-1.

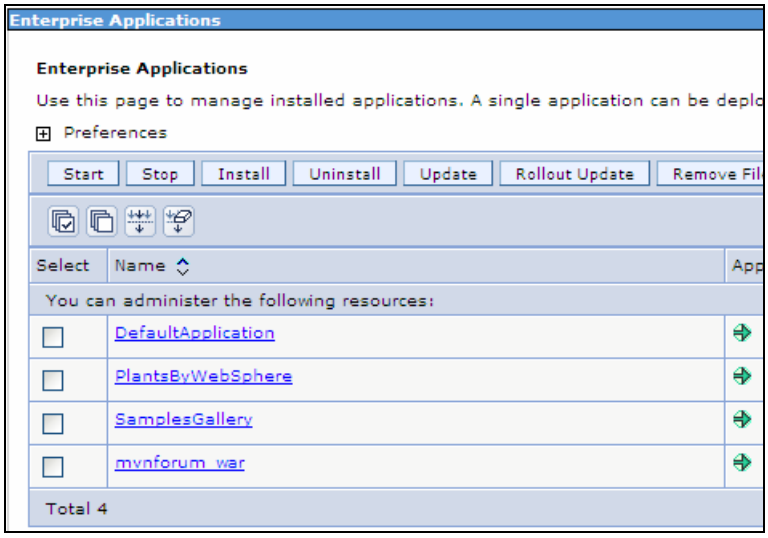


Figure 8-1 MvnForum deployed and started

2. Log in to the application using the following URL:

   `http://localhost:9080/mvnforum`

Testing the application

Now you are ready to test the application the same way you tested it when it was deployed to Tomcat.

We logged in to the application using the admin user, changed the avatar picture and email address. After that simple test, we checked the log files and did not find any errors.

Once again, this is not a complete test, and it is not recommended for a production environment, but it was good enough to know that the application was successfully migrated.

8.5 Easy JSP Forum migration

Easy JSP Forum is an Sun Public License licensed open source project hosted by SourceForge. This term of licence derives from an application that is created by a Sun application as NetBeans. The source has the approval by the Free Software Foundation (FSF) as a free software license, and by the Open Source Initiative (OSI) as an open source license. We can modify it and distribute your source code as we are used to under GPL license. Easy JSP Forum is a J2EE 1.4 application that demonstrates the capabilities of the J2EE platform.

Easy JSP Forum implements the JSP 2.0 and Servlet 2.4. In the distribution code the application connects to a Microsoft Access database under an ODBC data source. In this example, we migrate the Easy JSP Forum to a DB2 database under WebSphere Application Server V7 data source using type 4 driver. Easy JSP Forum was included in this book for the following reasons:

- It is based on popular open source frameworks.
- It was designed to run on multiple application servers.
- It is not specifically designed to run on WebSphere Application Server V7.
- It is not fully compliancy to DB2 backend systems.

Easy JSP Forum is an interesting migration example because it was implemented with several technologies and requires database adjusts and source code as well.

8.5.1 Migration approach

To migrate the application to WebSphere Application Server V7 we used WebSphere Application Server Toolkit V6.1, but you are not restricted with this. You can use other IDEs, as well

You have the option to migrate an application such as Easy JSP Forum to WebSphere Application Server V7 without using an IDE as Ant build scripts.

8.5.2 Configuring the source environment

This section describes all the required steps for setting up the source environment. Follow the instructions in 8.3, “Software installation” on page 277 to install the software needed in the source environment.

Easy JSP Forum was designed to run on Tomcat 5.x. To run it in Apache Tomcat 6.0.26 no extra steps are required, the distribution instructions shows examples in Tomcat 5.x, you should follow the steps exactly. We do not migrate to DB2 Universal Database V9.5 in the source environment.

Getting the Easy JSP Forum application

Before starting the migration, obtain the source files for the Easy JSP Forum application. You can download the `easyjspforum.zip` file with all source files from the following web page:

`http://sourceforge.net/projects/easyjspforum/files/`

In our scenario, we downloaded and extracted the application to the following directory (later referred to as `<source_home>`):

`C:\apache-tomcat-6.0.26\webapps\forum`

Configuring the database

This section describes how to configure the Easy JSP Forum application to access the Microsoft Access database supported by default. No changes are required in source code of application.

Follow these steps to configure Easy JSP Forum to use Microsoft Access database.

1. Go to **Start Settings** → **Control Panel** → **Administrative Tools** → **Data Sources**.
2. Click the System DSN tab of the ODBC dialog box.
3. Click **Add**. Select **Microsoft Access (*mdb)**, click **Finish**.
4. In Database area, click **Select**. Point it to:

   `<source_home>\database\Forum.mdb`
5. In the Data Source Name field, enter `Forum`.
6. Click **OK** twice to leave the ODBC Data Source Administrator window.

Deploying the application

As you copied the application distribution to the `C:\apache-tomcat-6.0.26\webapps\forum` folder, no deploy is necessary. You just need to follow the following steps to start the Tomcat server:

1. Open a command line and change the directory to `<tomcat_home>\bin`.
2. Start Tomcat by issuing the **`startup.bat`** command.

   You should see no errors in the log if the application was deployed successfully.
3. Access the application at the following URL:

   `http://localhost:8080/forum/`

Testing the application

The Easy JSP Forum has no automated tests implemented with JUnit, JUnitEE, HttpUnit or any other automated test tool. Therefore, we have to test the application manually.

In a simple test, we browsed the application, signed in using `admin` as both user and password, and edited the members (tasks such as changing the last name and email fields). We created a new forum and posted threads on it and replies to the same thread. The current admin profile was changed also. Those simple procedures took us through the application with no errors. We checked the log files and did not find any error or warnings.

At this point, we have successfully built, deployed, and tested Easy JSP Forum into Apache Tomcat 6.0.26, which is its original environment. Having the application working properly on the source environment is our starting point for migrating into WebSphere Application Server V7.

8.5.3 Migrating the Easy JSP Forum application to WebSphere

This section describes the necessary steps for migrating the application from Tomcat to WebSphere Application Server. The steps of this migration involve DB2 Universal Database V9.5 database creation, WebSphere Application Server V7 data source configuration, application import to WebSphere Application Server Toolkit V6.1, source code change to connect to WebSphere data source, and sql syntaxes fixes.

In this section, you modify the build files and customize them to prepare the application to run in WebSphere Application Server V7. After you have those steps completed, the application is ready to be built and deployed in WebSphere Application Server V7.

The last part of this section describes how to configure the resources required by the Easy JSP Forum to run in WebSphere Application Server V7, and how to deploy and test the migrated application.

Creating the DB2 database

The Easy JSP Forum comes with Microsoft Access database. The DB2 Universal Database V9.5 has been chosen as database backend to this migration approach. There is no database creation script provided with source distribution. In our scenario we decided to create the database and tables manually. You can also use tools to export the Access database to DB2. There is a developerWorks article, *Migrating a Microsoft Access 2000 Database to IBM DB2 Universal Database 7.2* that provides additional information.

The article can be found at the following web page:

http://www.ibm.com/developerworks/data/library/techarticle/alazzawe/0112alazzawe2.html

Follow the steps shown in Example 8-5 to create the database and the tables manually.

Example 8-5 Creating the database and the tables

```
db2 create db EASY
db2 connect to EASY user fxavier using your_password
------------------------------------------------
-- Creation commands for table "FXAVIER "."BOXES"
------------------------------------------------
DB2 CREATE TABLE "FXAVIER "."BOXES"  (
        "BOX_NAME" VARCHAR(100) ,
        "SORT_DESC" VARCHAR(100) ,
        "BOX_ID" INTEGER ,
        "MEMBER_ID" INTEGER )
       IN "USERSPACE1" ;
------------------------------------------------
-- Creation commands for table "FXAVIER "."LEVELS"
------------------------------------------------
DB2 CREATE TABLE "FXAVIER "."LEVELS"  (
        "GRADE" VARCHAR(100) ,
        "MIN_POST" INTEGER ,
        "MAX_POST" INTEGER ,
        "DESCRIPTION" VARCHAR(100) )
       IN "USERSPACE1" ;
------------------------------------------------
-- Creation commands for table "FXAVIER "."MEMBERS"
------------------------------------------------
DB2 CREATE TABLE "FXAVIER "."MEMBERS"  (
        "MEMBER_ID" BIGINT NOT NULL ,
        "USERNAME" VARCHAR(100) NOT NULL ,
        "PASSWORD" VARCHAR(100) NOT NULL ,
        "FIRSTNAME" VARCHAR(100) NOT NULL ,
        "LASTNAME" VARCHAR(100) NOT NULL ,
        "EMAIL" VARCHAR(100) NOT NULL ,
        "REGDATE" VARCHAR(100) NOT NULL ,
        "TYPE" VARCHAR(100) NOT NULL )
       IN "USERSPACE1" ;
------------------------------------------------
-- Creation commands for table "FXAVIER "."THREADS"
------------------------------------------------
```

```
DB2 CREATE TABLE "FXAVIER "."THREADS"  (
        "THREAD_ID" BIGINT NOT NULL ,
        "BOX_ID" BIGINT NOT NULL ,
        "PARENT_ID" BIGINT NOT NULL ,
        "MEMBER_ID" BIGINT NOT NULL ,
        "SUBJECT" VARCHAR(100) NOT NULL ,
        "POST_TEXT" VARCHAR(1000) NOT NULL ,
        "POST_DATE" BIGINT NOT NULL )
       IN "USERSPACE1" ;
```

Now the table Members needs to be populated to log in the application. Run the commands in Example 8-6 in the DB2 Command Window.

Example 8-6 Populate the table Members

```
db2 connect to EASY user fxavier using your_password
db2 INSERT INTO FXAVIER.MEMBERS VALUES
(1,'admin','admin','Forum','Admin','fxavier@br.ibm.com','Fri May 07
10:44:38 ICT 2010','Administrator')
db2 INSERT INTO FXAVIER.MEMBERS VALUES
(2,'mod','mod','Forum','Moderator','dt@test.com','Fri May 07 10:44:38
ICT 2010','Moderator')
db2 INSERT INTO FXAVIER.MEMBERS VALUES
(3,'member','member','Forum','Member','dt@test.com','Fri May 07
10:44:38 ICT 2010','Member')
```

The DB2 database setup is now complete.

Configuring the data source in WebSphere

Because Easy JSP Forum requires a database, you need to provide that database configuration in the application server.

Perform the following steps to configure the data source from the WebSphere Administrative Console.

1. Open the WebSphere Administrative Console:

   `http://localhost:9060/ibm/console`

2. Log in and click **Resources** → **JDBC** → **JDBC Providers** → **New** to create a new JDBC provider with the characteristics shown in Table 8-6, and click **Next**.

Table 8-6 JDBC provider values

Parameter	Value
Database type	DB2
Provider type	DB2 Universal JDBC Driver Provider
Implementation type	Connection pool data source
Name	DB2 Universal JDBC Driver Provider

3. Specify the correct path to the DB2 JDBC driver and click **Apply**. In our migration example, the path to the JDBC driver is `C:\Program Files\IBM\SQLLIB\java`. Click **Finish**, then save the configurations to the master configuration.
4. Click **Resources** → **JDBC** → **Data Sources**, select the scope and click **New**. Create a new data source with the characteristics in Table 8-7 and click **Next**.

Table 8-7 Data Source values

Parameter	Value
Data source name	EASY
JNDI Name	jdbc/easy

5. Click the **Select an existing JDBC provider** radio button and select **DB2 Universal JDBC Driver Provider**. Click **Next**.

6. In the “Enter database specific properties for the data source” panel, enter the values shown in Table 8-8, and click **Next**.

Table 8-8 Database specific properties

Parameter	Value
Driver type	4
Database name	EASY
Server name	ibm-6474l1bb23c
Port number	50000
Use this data source in container managed persistence (CMP)	Checked

7. On the Setup security aliases window, click **Next**. Review the summary in next window and click **Finish**.
8. Return to the data source you just created, Click **JAAS - J2C authentication data** in the Related Items panel, and click **New**.
9. Create a new alias, specify the values shown in Table 8-9, and click **OK**.

Table 8-9 J2C alias configuration

Parameter	Value
Alias	JAASEASY
UserId	fxavier
Password	your_password

10. Return to the data source you just created. From the Component-managed authentication alias drop-down menu, select the J2C alias you just created. Click **OK** and save your configuration.
11. Verify the connection to the database. Select the data source you just created from the list and click **Test connection**.

Import the application to AST 6.1

The Easy JSP Forum is not distributed using the WAR of EAR specifications and is not compatible with the folder standards of this specification. To deploy it to WebSphere Application Server V7 we have to import and package it as EAR application. We need also to modify the source code to access the data source.

Open the WebSphere Application Server Toolkit V6.1 and follow these steps:

1. Create a new workspace.
2. Go to **File** → **New Project**. Expand the web option and select **Dynamic Web Project**. Click **Next**.
3. Enter a project name. We entered `forum` as the project name. Leave the other options as default. Click **Finish**.
4. In the J2EE perspective right-click the forum folder and **Import** → **Import**. Expand the General list and select **File System** and click **Next**.
5. Select all the folders under the forum folder, as seen in Figure 8-2, and click **Finish**.

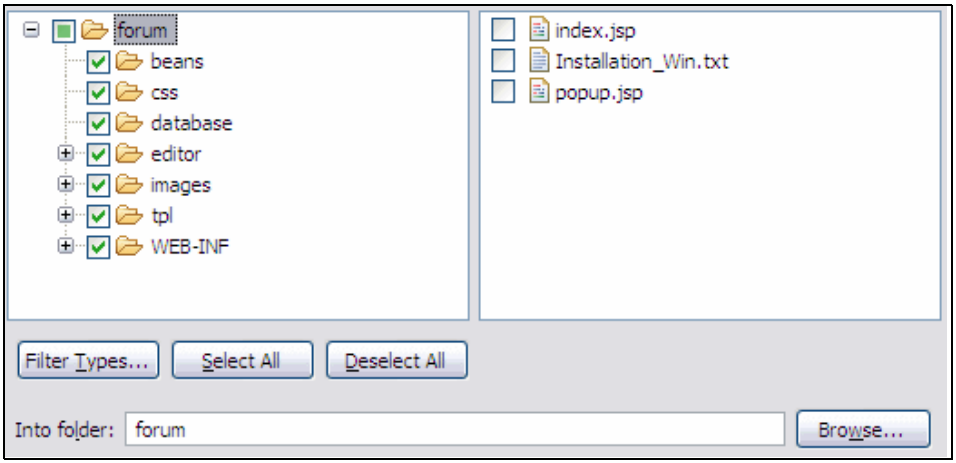


Figure 8-2 Files selection from source_home. Do not select the forum folder.

The application has now been successfully imported to WebSphere Application Server Toolkit V6.1. However, the Easy JSP Forum code is not prepared to use data sources. A source code modification is required to access the DB2 database. The application contains three classes that perform all operations:

- Members.java
- Threads.java
- Boxes.java

Add the snippet in Example 8-7 before these class definitions.

Example 8-7 Adding the required packages to access the data source

```
import javax.naming.*;
import javax.sql.*;
```

To connect to the data source you need to change the connection parameters as follows. Example 8-8 shows how you should perform this operation.

Example 8-8 Changing the connection parameters

```
Comment out all entries of the following lines:
//       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//       Connection con =
DriverManager.getConnection("jdbc:odbc:Forum");import javax.sql.*;

Add the following lines after comment out part:
Context ctx=new InitialContext();
DataSource ds=(DataSource)ctx.lookup("jdbc/easy");
Connection con=ds.getConnection();
```

During our tests SQL syntaxes did not work when accessing the DB2 database. You need to fix queries so that the application does not show an exception. The problem in our case was the square brackets symbol between the fields. Example 8-9 shows how we fixed this problem.

Open the files and remove the square brackets in the following SQL syntaxes:

Example 8-9 Remove the square brackets in the following SQL syntaxes

```
File Boxes.java
Line 121 : String query = "INSERT INTO Boxes (box_id, member_id,
[box_name], [sort_desc]) " +

File Members.java
Line 75 : String query = "SELECT member_id FROM Members WHERE
[username]='" + username + "'";
Line 133 : String query = "INSERT INTO Members (member_id, [username],
[password], [firstname], [lastname], [email], [regdate], [type]) " +
Line 166 :        "email='" + email + "' WHERE [username]='" + username
+ "'";
Line 211 : String query = "SELECT * FROM Members WHERE [username]='" +
username + "'";

File Threads.java
Line 220 : String query = "INSERT INTO Threads (thread_id, parent_id,
box_id, member_id, [subject], [post_text], post_date) " +
Line 244 : String query = "INSERT INTO Threads (thread_id, parent_id,
box_id, member_id, [subject], [post_text], post_date) " +
```

Building the application

All the changes needed to deploy the application in WebSphere Application Server V7 are complete. To build the EAR file perform the following steps:

1. Inside the WebSphere Application Server Toolkit V6.1 workbench in J2EE view, right-click in the forumEAR folder.
2. Go to **Export** → **EAR File**.
3. Choose a folder destination and click **Finish**.

Deploy the exported EAR file in WebSphere Application Server.

Deploying the application

Perform the following steps to deploy the application:

1. From the WebSphere Administrative Console, click **Applications** → **New Application** → **New Enterprise Application**.
2. Specify the path to the EAR file you just created and the context root as described in Table 8-10 and click **Next**.

Table 8-10 Installation values

Parameter	Value
Local File System Path	<source_home>\forumEAR.ear
Context Root	/forum

3. Select the **Fast Path - Prompt only when additional information is required** check box.

   Expand **Choose to generate default bindings and mappings**, and select the **Generate Default Bindings** check box. Click **Next**.
4. In the Step 1: Select installation options window, leave all the default options and click **Next**.
5. In the Step 2: Map modules to servers window, leave all the default options and click **Next**.
6. In the Step 3: Summary window, click **Finish**. It might take a few minutes to deploy the application and validate all the mappings. After you are finished, save to the master configuration and synchronize the nodes.

Next, perform the following steps to start the application.

1. From the WebSphere Administrative Console, click **Applications** → **Application Types** → **Enterprise Applications**. Select the **forumEAR** application check box and click **Start**.

   You should see the application started and a green arrow to the right of the application name, as in Figure 8-3.

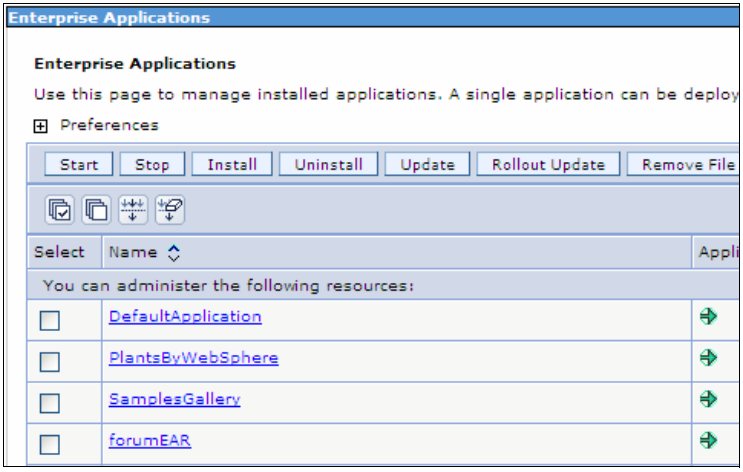


Figure 8-3 Easy JSP Forum deployed and started

2. Log in to the application using the following URL:

   `http://localhost:9080/forum`

Testing the application

Because we do not have any kind of automated test, we have to test the application manually, following the same steps we performed to test the application running in Tomcat. Again, we signed in using `admin` as user name and password, and edited the members (tasks such as changing the last name and email fields).

We created a new forum and posted threads on it and replied to the same thread. The current admin profile was changed also. This simple procedure take us through the application with no errors. We checked the log files and did not find any error or warning. The results were the same, so we can say that the application was successfully migrated.

9

Migrating from GlassFish

This chapter describes the migration process and the individual migration issues and solutions found when migrating two Java EE 5 applications from GlassFish Server Open Source Edition 2.1.1 (or Oracle GlassFish Server 2.1.1, if you are using the commercial edition) to WebSphere Application Server V7.

By reading and following the steps described here, you can deploy a complete application, which was written for GlassFish, in a WebSphere Application Server environment.

Because the compatibility of application servers nowadays is surprisingly good, and also, due to the Java EE standards evolutions, the migrations from a pure Java EE 5 applications should not be that difficult and should not raise complex code translations and changes.

This chapter has the following sections:

- "Introduction" on page 302
- "Prerequisites and assumptions" on page 302
- "Java Pet Store Reference application" on page 303
- "CalculatorWS: A sample web services application" on page 318
- "Major issues when migrating web services applications" on page 322
- "Summary" on page 324

© Copyright IBM Corp. 2010. All rights reserved.

9.1 Introduction

GlassFish is an open source application server project led by Sun Microsystems for the Java EE platform. The proprietary version is called Sun GlassFish Enterprise Server. GlassFish is free software, dual-licensed under two free software licences: the Common Development and Distribution License (CDDL) and the GNU General Public License (GPL) with the classpath exception.

For more information about GlassFish licensing information, see the following web site:

https://glassfish.dev.java.net/public/CDDL+GPL.html

GlassFish is based on source code released by Sun and Oracle Corporation's TopLink persistence system. It uses a derivative of Apache Tomcat as the servlet container for serving web content, with an added component, called *Grizzly*, which uses Java NIO.

Nowadays, there are two distribution packages for GlassFish Application Server:

- GlassFish Server Open Source Edition
- Oracle GlassFish Server commercially supported by Oracle.

For more information about GlassFish Application Server, its packaging formats and distributions and versions, visit the following web pages:

- https://glassfish.dev.java.net/public/downloadsindex.html
- https://glassfish.dev.java.net/public/comparing_v2_and_v3.html

9.2 Prerequisites and assumptions

The first application, Java Pet Store, is documented as a Java EE 5-compliant web module.

For migration purposes, we decided to use GlassFish OpenSource Edition version 2.1.1, because it is fully compliant with Java EE 5. The Java Pet Store was then built and deployed on GlassFish 2.1.1.

The second application is a simple web services application, also JavaEE 5 compliant, which was developed by the authors of this book. This application was created to point out important considerations when migrating web services applications.

To develop and deploy this second application, we are using Netbeans IDE 6.8 bundled with GlassFish V3 (build 74.2). Our test environment is an Intel-based Windows 2003 Server.

Using a stable version of GlassFish and WebSphere is always recommended. For that reason, keep in mind that at the time of the publication, it can be available in various versions or in fixes for those products.

9.3 Java Pet Store Reference application

For this first migration, we are using a common and widespread application: the Java Pet Store 2.0 Reference application from Sun Microsystems. This application can be found at the Java BluePrints web site:

https://blueprints.dev.java.net/petstore/

To download Java Pet Store Reference Application, see the following web page:

https://blueprints.dev.java.net/servlets/ProjectDocumentList?folderID=5315&expandFolder=5315&folderID=0

This application demonstrates the use of various features of the Java EE 5 platform:

- JavaServer Faces

  Many of the AJAX features are implemented as reusable JavaServer Faces components.

- Java Persistence API

  The application uses Java Persistence APIs to create an object/relational mapping layer.

- Dependency injection

  The application uses dependency injection instead of deployment descriptors.

These features make the migration process easier than earlier applications due to the Java EE 5 standards, and also due to the fact that it does not use vendor-specific features. The value of using Java EE standards is important when writing portable applications. For more information about developing portable applications, refer to the Appendix A, “Development practices for portable applications” on page 325.

The Pet Store application comes with full source-code available under a BSD-style license.

Figure 9-1 gives an overview of the installation and migration paths used for the Java Pet Store 2.0 Reference application.

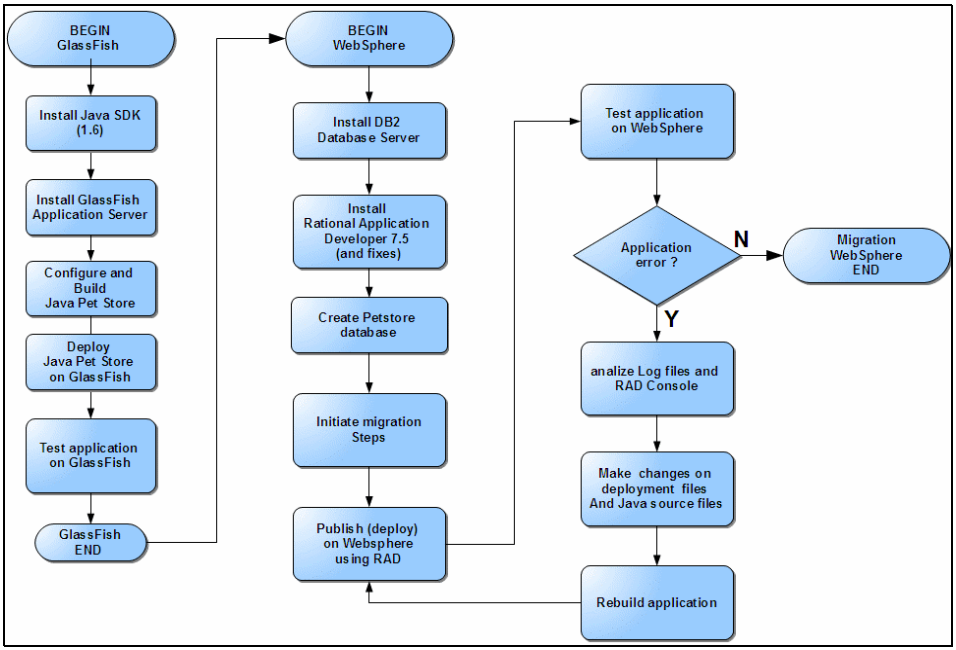


Figure 9-1 Java Pet Store migration steps overview

9.3.1 Installing and configuring GlassFish V2.1.1

Perform the following steps for installing and configuring the GlassFish:

1. Download the SDK 6 from the following web page:

   `http://java.sun.com/javase/downloads/index.jsp`

2. Install Java SDK 6. See the Sun Java documentation at the following web page for more information:

   `http://java.sun.com/javase/6/webnotes/install/index.html`

   Important: Make sure that you create the correct JAVA_HOME and PATH entries in your environment variables.

3. Download the GlassFish Application Server from the following web page:

   `https://glassfish.dev.java.net/public/downloadsindex.html`

4. Install GlassFish by issuing the commands in Example 9-1.

Example 9-1 GlassFish installation commands

```
# java -Xmx256m -jar <glassfish filename>.jar
# cd <\GlassFish install home dir>
# lib\ant\bin\ant -f setup.xml
```

Important: Make sure to change the folders and filespec (text inside brackets), according to your environment.

5. The build process should end with a `BUILD SUCESSFULL` message.

   For more information about GlassFish installation, see the document at the following web page:

   http://docs.sun.com/app/docs/doc/821-0187/abrar?a=view

6. Issue the commands in Example 9-2 to start GlassFish.

Example 9-2 GlassFish initialization commands

```
# cd <\GlassFish install home dir>
# bin\asadmin start-database
# bin\asadmin start-domain
```

7. (Optional) Start the GlassFish Admin Console:

   `http://localhost:4848`

   Use the default username: `admin` and the default password: `adminadmin`.

9.3.2 Building and deploying Java Pet Store on GlassFish

Perform the following steps to install the Java Pet Store application on Glassfish.

1. Download Java PetStore Application from the following web page:

   https://blueprints.dev.java.net/servlets/ProjectDocumentList?folderID=5315&expandFolder=5315&folderID=0

2. Unpack the JAR file using the commands in Example 9-3.

Example 9-3 Unpack commands

```
# java -Xmx256m -jar <java petstore filename>.jar
# cd <glass fish install folder>
```

3. Build Java Pet Store for GlassFish. To do this, modify the properties inside the `<petstore home>/bp-project/build.properties` file, according to your environment and then issue the following commands:

   ```
   # <glassfish folder>\lib\ant\bin\ant setup
   # <glassfish folder>\lib\ant\bin\ant run
   ```

   Just like the other one, the BUILD and RUN commands must end with the `BUILD SUCESSFULL` message.

For more information about Java Pet Store building and installation for GlassFish, see `index.html` within the application JAR file or the Java PetStore official website (see 9.2, “Prerequisites and assumptions” on page 302).

9.3.3 Migration to WebSphere Application Server

When trying to migrate any Java EE application, we could take one of two approaches. The first one is just to deploy the EAR files built for GlassFish (with ant) straight into WebSphere Application Server. Such an approach, which is easier, works only with pure Java EE applications, with no vendor-specific features and configuration files. Due to the differences in deployment and configuration files, such strategy might fail. In fact, for Java Pet Store application, that was what happened.

The second approach, which was used in this book, is to bring the Java source files and the deployment and configuration files to the Rational Application Developer, and work on these files looking for non-supported directives and also non-standards in Java code.

The following sections describe each step to perform on descriptors files and on the Java source code files to migrate the whole application to WebSphere Application Server V7.

Configuring the initial environment

Perform the following steps to configure the initial environment:

1. Install DB2 according to the instructions in “IBM DB2 Server Express Edition” on page 97.
2. Create a database for Pet Store application by performing the following steps:
   a. Open the DB2 Command Editor for DB2 (Figure 9-2) and issue the # **`create database petstore;`** command to create the database.

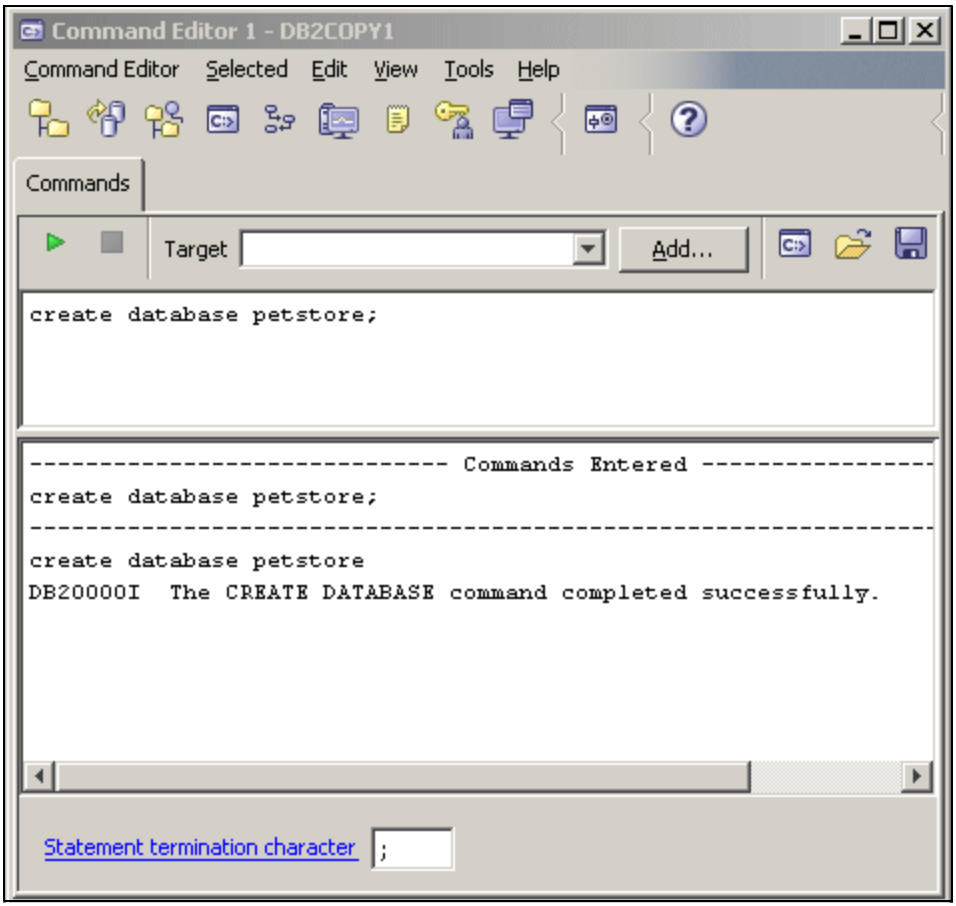


Figure 9-2 Creating “petstore” database in DB2

   b. After the database is created, you can connect to it, create the tables, and populate the petstore database. Using DB2 Command Editor, issue the **`connect to <dbname> user <username> using <userpswd>;`** command.

c. Open the \<petstore home folder>\setup\sql\javadb\petstore.sql file.

d. Execute the list of commands, as shown in Figure 9-3.

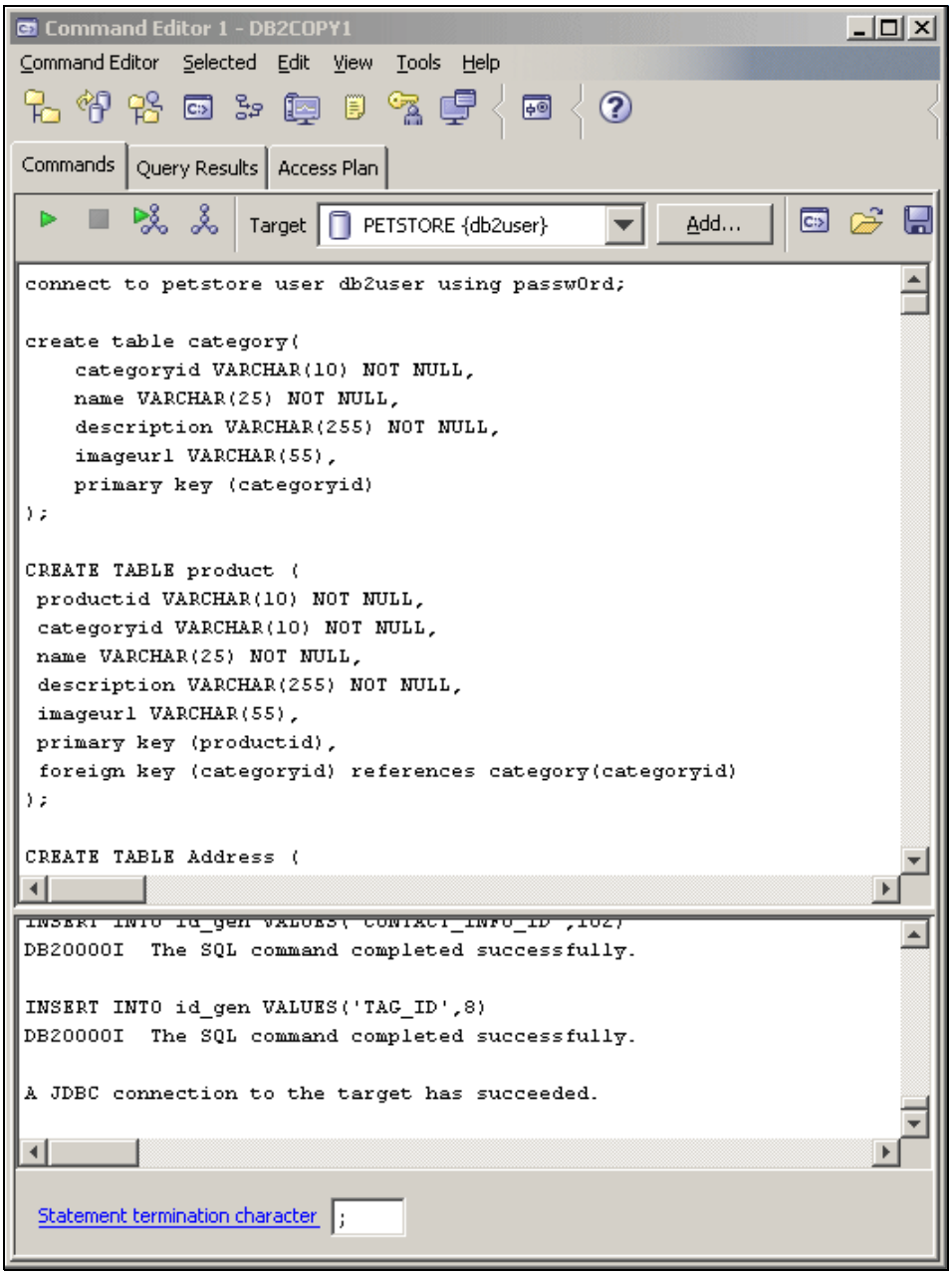


Figure 9-3 Creating tables and populating "petstore" database.

Note: For more information about DB2 commands, see the following web site:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0406dang/index.html

Migrating the sample application

As already stated before, to migrate the Java Pet Store application, we needed to work on descriptors files and Java source code.

The following steps help us to understand how one can import Java source code libraries, configuration, and descriptors files using IBM Rational Application Developer V7 (RAD) to make things easier. Then we initiate the application migration to our WebSphere Application Server test environment, again using RAD to facilitate and make use of the process.

Perform the following steps to initiate the migration of the Java Pet Store application:

1. Install IBM Rational Application Developer for WebSphere v7.5, according to the instructions found in the “IBM Rational Application Developer for WebSphere Software” on page 88.
2. Open RAD, select your workspace folder, and click **OK**. See Figure 9-4.

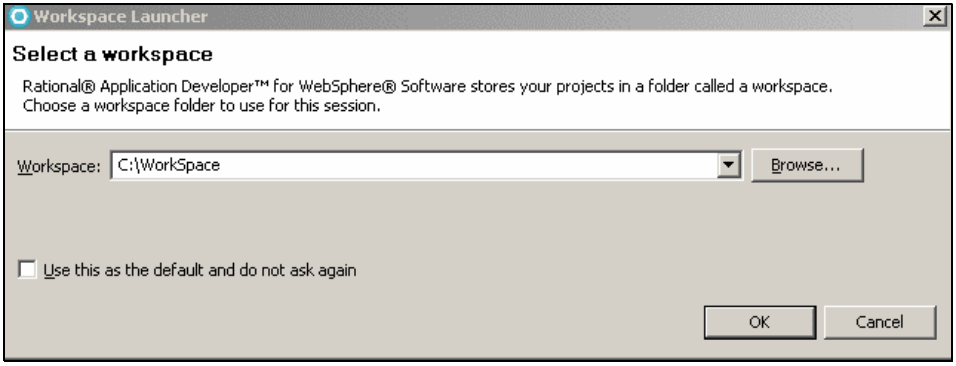


Figure 9-4 Select the workspace folder according to your environment

3. Reset your perspective by choosing the Java EE perspective:
   a. Select **Window** → **Open Perspective** → **Other**.
   b. Select **Java EE** and click **OK**. See Figure 9-5.

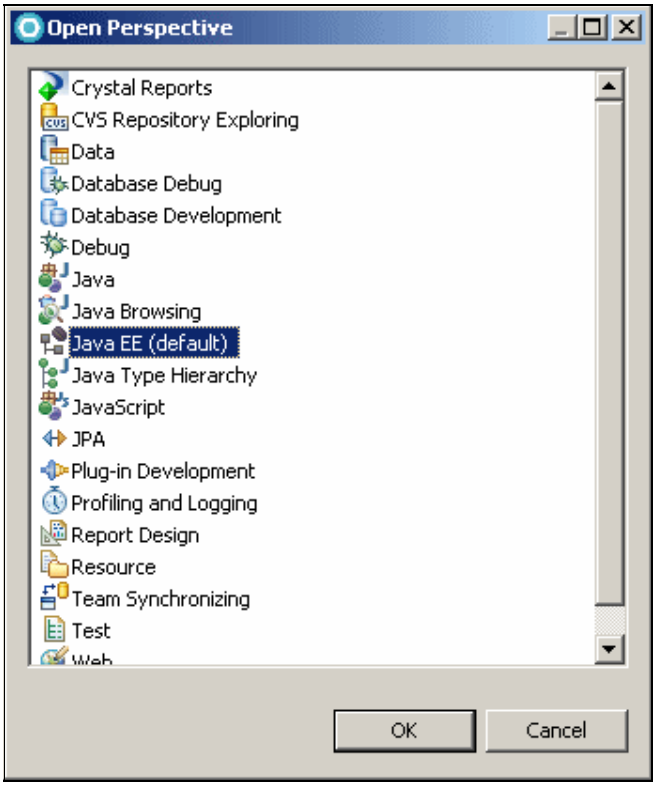


Figure 9-5 Resetting the RAD perspective to a Java EE Project look and feel

4. Create a dynamic web project. Click **File** → **New** → **Dynamic Web Project**.

   Select the name and general properties of the project, according to your environment. See the examples shown in Figure 9-6 and Figure 9-7 on page 312.

   Observe the Context Root and EAR Project Name fields. They define properties and descriptors file and affect the way your application is deployed on WebSphere Application Server and how users access the web interface for Pet Store application.

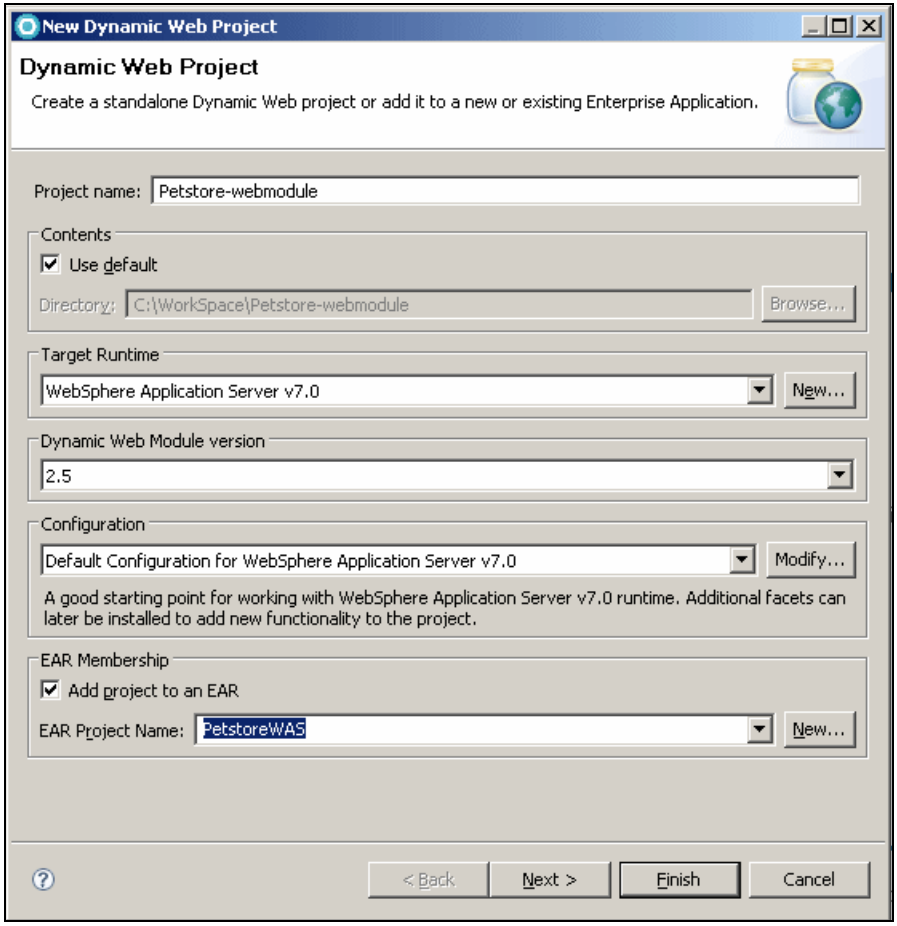


Figure 9-6 Creating a dynamic web project #2

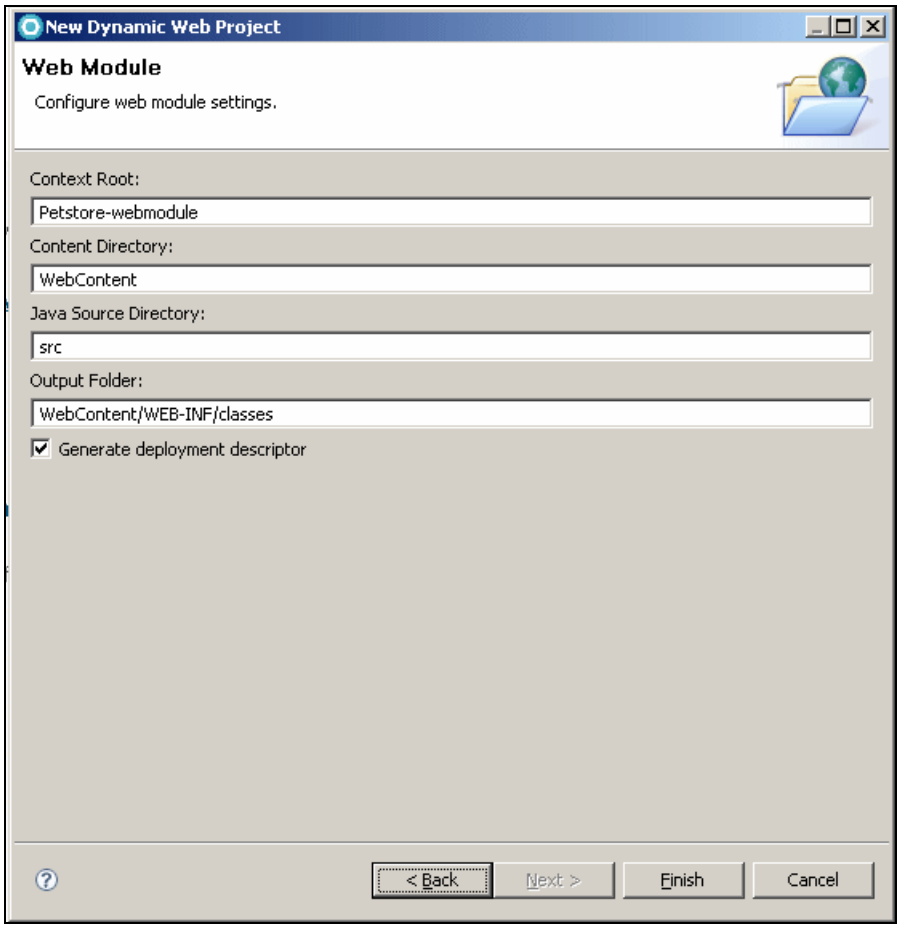


Figure 9-7 Creating a dynamic web project #2

5. After you create a project, copy the source files from the Java Pet Store original distribution to the RAD Workspace:

   From `\<petstore app folder>\src\*`

   To `<workspace path>\<web module name>\src`

6. Copy the JAR repository and external classes library from the Java Pet Store distribution to the RAD Workspace:

   From `\<petstore app folder>\lib\*`

   To `<RAD workspace path>\<webodule name>\WebContent\WEB-INF\lib`

   There will be red marks at the Java files, showing that files cannot be found. This situation occurs, because the directory structure for GlassFish, which is slightly different to WebSphere. See Figure 9-8.

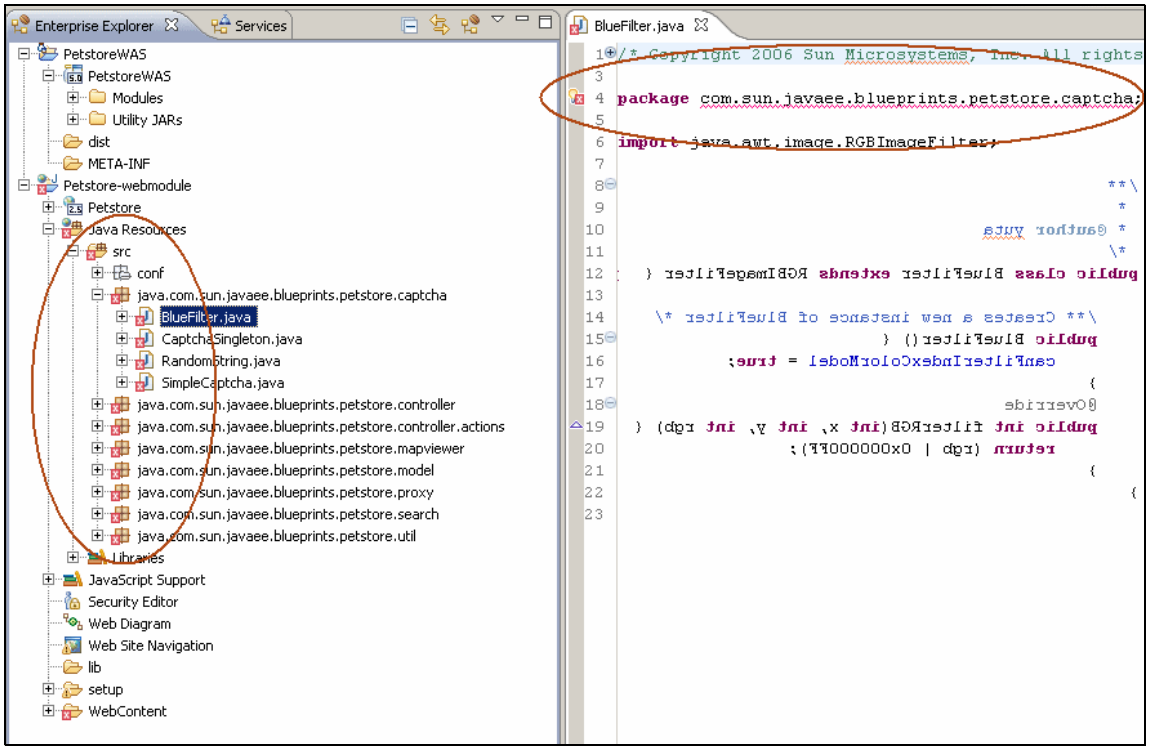


Figure 9-8 Package name problem when importing to RAD.

To solve this issue, change the location of the source files inside RAD workspace:

From `<workspace path>\<web module name>\src\java\*`

To `<workspace path>\<web module name>\src\`

In summary, move the com folder and all its children to the parent folder.

The directory structure should look like the list in Example 9-4 on page 314.

Example 9-4 Folder structure for source code

```
<workspace>\<web module>\src\com
<workspace>\<web module>\src\com\sun
<workspace>\<web module>\src\com\sun\javaee
<workspace>\<web module>\src\com\sun\javaee\blueprints
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\captcha
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\controller
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\mapviewer
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\model
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\proxy
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\search
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\util
<workspace>\<web module>\src\com\sun\javaee\blueprints\petstore\controller\actions
```

The next step, is related to JSF and DOJO library used in this application for the VIEW layer (see more about MVC in “Use MVC” on page 331.

7. Extract the `ui.tld` file from `bp-ui-5.jar` library, issuing the following command:

   ```
   # jar -xvf <workspace>\<web
   module>\WebContent\WEB-INF\lib\bp-ui-5.jar META-INF/ui.tld
   ```

8. Move the extracted `ui.tld` file to `<workspace>\<web module>\WebContent\WEB-INF`.

   After these steps, the Dojo errors should disappear from RAD environment.

Where to look for more information: For more information about the Dojo Toolkit refer to the following web site:

http://www.dojotoolkit.org/

9. Create a resource reference in the WebSphere Web Bindings Descriptor (`ibm-web-bnd.xml`) by performing the following steps:

   a. Right-click the web module project and select **Java EE** → **Generate WebSphere Bindings Deployment Descriptor**.

      This action should open the design tab of the Web Bindings Editor.

   b. Click **Add** and select **Resource Reference**. Enter the following values:

      Name: `jdbc/PetstoreDB`

      Binding Name: `jdbc/PetstoreDB`

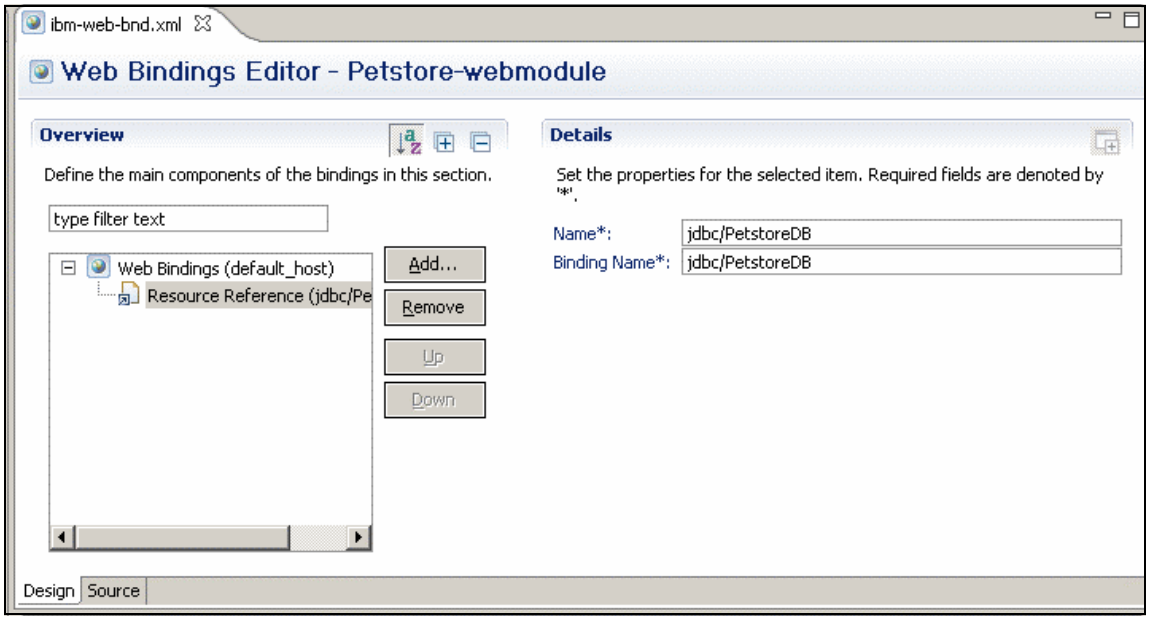


Figure 9-9 Create a Resource Reference for a JDNI name

The JPA result list is read-only.

The getResultList() implementation method in WebSphere Application Server, returns a un-modifiable list according to the OpenJPA implementation, but apparently this does not occur in the GlassFish Application Server implementation.

During our migration steps and, by debugging the error messages in WebSphere log files, it was possible to realize that part of the code in the Java Pet Store application tries to sort the result list returned by a JPA query. This is shown in Example 9-5 on page 316.

Example 9-5 Original index.jsp raises an "Unsupported Operation" exception due to read-only result list from JPA

```
<%
try {
    CatalogFacade cf =
(CatalogFacade)config.getServletContext().getAttribute("CatalogFacade")
;
    List<Tag> tags=cf.getTagsInChunk(0, 12);
    // since top 20 come from database or desending refCount order,
need to reorder by tag name
    Collections.sort(tags, new Comparator() {
        public int compare(Object one, Object two) {
             int cc=((Tag)two).getTag().compareTo(((Tag)one).getTag());
             return (cc < 0 ? 1 : cc > 0 ? -1 : 0);
        }
    });
%>
```

To solve this error and avoid changes on persistent objects, we implemented changes in the source code of the `index.jsp` file, as shown in Example 9-6.

Example 9-6 Modified index.jsp

```
<%
try {
    CatalogFacade cf =
(CatalogFacade)config.getServletContext().getAttribute("CatalogFacade")
;
    List<Tag> tagsRO = cf.getTagsInChunk(0, 12);

    List<Tag> tags = new ArrayList<Tag>() ;
    tags.addAll(tagsRO) ;

   // since top 20 come from database or desending refCount order, need
to reorder by tag name
    Collections.sort(tags, new Comparator() {
        public int compare(Object one, Object two) {
             int cc=((Tag)two).getTag().compareTo(((Tag)one).getTag());
             return (cc < 0 ? 1 : cc > 0 ? -1 : 0);
        }
    });
%>
```

GlassFish ignores double slashes in the URLs and by that, it can still parse the URL requests, but WebSphere Application Server is more sensitive to these malformed URLs.

For example, the URL `http://app1.itso.ibm.com//app_context`, causes a HTTP error in WebSphere Web Container, due to the double slashes (“//”) after server and domain names.

The development team of Java Pet Store application included an extra slash “/” when building a baseURL string variable, but it is unnecessary, because the `javax.servlet.http.HttpServletRequest.request.getContextPath()` method already returns the slash character “/”.

According to the Java Servlet API Documentation, *“The path starts with a "/" character but does not end with a "/" character”.*

See the following web page for more information:

`http://java.sun.com/products/servlet/2.5/docs/servlet-2_5-mr2/javax/servlet/http/HttpServletRequest.html#getContextPath%28%29`

Example 9-7 shows the Original CatalogXmlAction.java (line 70 in the source code).

Example 9-7 Original CatalogXmlAction.java (line 70)

```
String baseURL = "http://" + request.getServerName() + ":" +
request.getServerPort() + "/" + request.getContextPath() +
"/ImageServlet/";
```

Example 9-8 shows our changes.

Example 9-8 New CatalogXmlAction.java (line 70)

```
String baseURL = "http://" + request.getServerName() + ":" +
request.getServerPort() + request.getContextPath() +
"/ImageServlet/";
```

This change in CatalogXmlAction.java avoids HTTP 404 errors in WebSphere Application Server when building the image URLs and the Pet Store catalog.

By running those steps and using the RAD to assist you during migration, it should not be difficult to migrate this sample application. As already stated, the Java EE standards are evolving in a way that all applications will become more compatible and portable among platforms and application server vendors.

9.4 CalculatorWS: A sample web services application

The CalculatorWS is a web services sample application developed by the authors of this book to point out the issues that you can encounter when migrating web services applications to WebSphere Application Server.

It is a simple enterprise application, compliant with Java EE 5 standards with only two operations: a multiplication and a factorial.

The application was initially written using Netbeans IDE 6.8 and deployed to a GlassFish V3 Application Server. The migration process was performed using the Rational Application Developer and deployed to the WebSphere Application Server.

To test the web services before and after the migration, we developed a simple client consumer application, which connects to the web service and requests results for both operations.

The source code of these sample applications can downloaded from the ITSO Redbooks website. See Appendix C, “Additional material” on page 375 for instructions on how to download this code.

Figure 9-10 on page 319 gives an overview of the development and migration paths used for the CalculatorWS, a sample web services application.

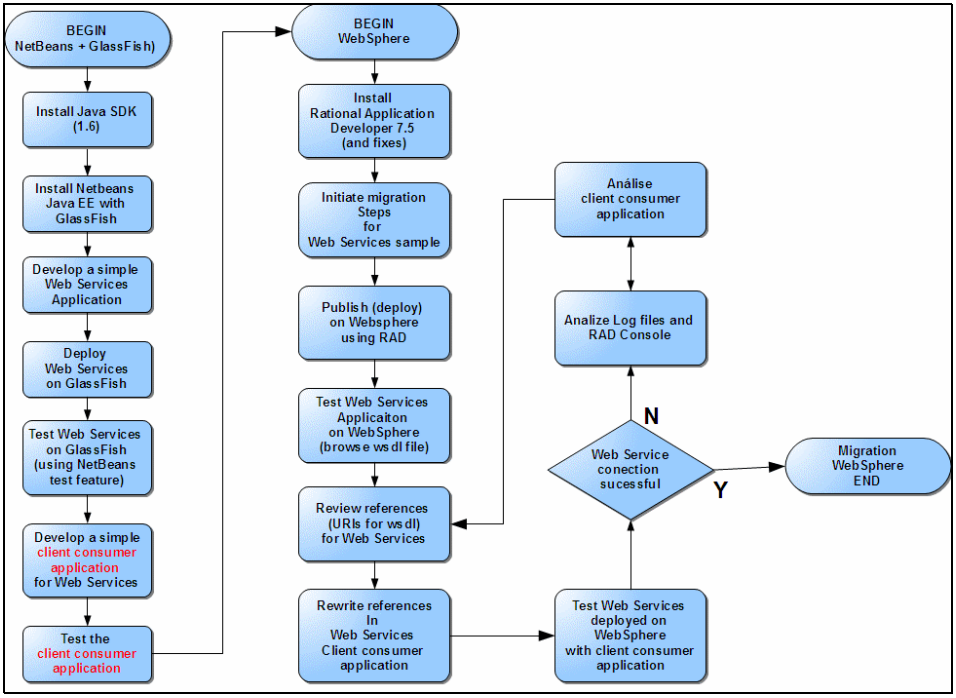


Figure 9-10 Sample web services application development and migration steps

9.4.1 Running CalculatorWS on NetBeans 6.8 and GlassFish 3

Perform the following steps to install NetBeans IDE 6.8 with GlassFish V3 followed by a deployment of CalculatorWS on Glassfish:

1. Download the Java SDK 6 from the following web page:

   `http://java.sun.com/javase/downloads/index.jsp`

2. Install Java SDK 6. See the Sun Java documentation for more information:

   `http://java.sun.com/javase/6/webnotes/install/index.html`

   Important: Make sure that you create the correct JAVA_HOME and PATH entries in your environment variables.

3. Download NetBeans installation file from the NetBeans web site:

   `http://netbeans.org/downloads/index.html`

   Select the complete version of the Java version (with Java EE Platform and GlassFish Enterprise Server).

4. Install NetBeans 6.8 Java EE bundle according to the installation guide, and according to your operating system and platform:

   http://netbeans.org/community/releases/68/install.html

5. Download the sample application CalculatorWS from (NetBeans Project) using the instructions provided in Appendix C, “Additional material” on page 375.
6. Copy the contents of these files to your NetBeans workspace.
7. (Optional) If you want to test the sample applications on GlassFish, perform the following steps:

   a. Right-click the **CalculatorWSApplication** project and select **Deploy**.

   b. Right-click **ClientConsumerForCalculatorWS** and select **Run**.

   c. Check the GlassFish console for any relevant messages.

9.4.2 Migration to WebSphere Application Server

The migration steps for CalculatorWS Web Service application is pretty simple, because you have the source code. The main step is copy the only Java source file into RAD workspace.

1. Install IBM Rational Application Developer for WebSphere v7.5, using the instructions found in “IBM Rational Application Developer for WebSphere Software” on page 88.
2. Open RAD and select your workspace folder and click **OK**, as shown in Figure 9-11.

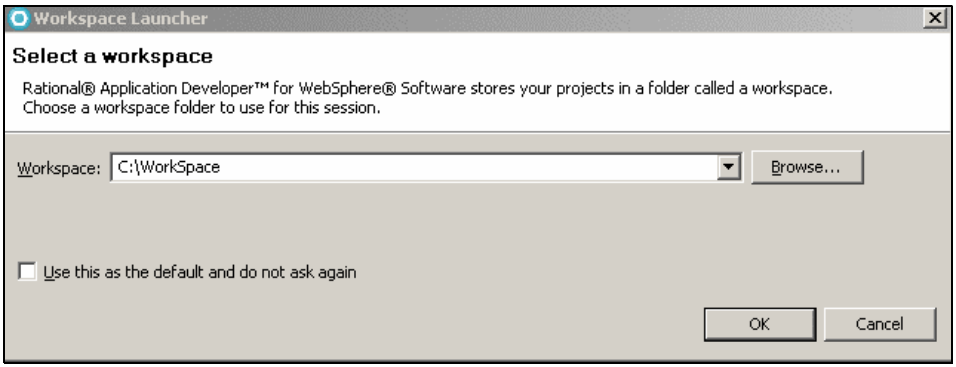


Figure 9-11 Select the workspace folder according to your environment

3. Reset your perspective by choosing the Java EE perspective:
   a. Select **Window** → **Open Perspective** → **Other**.
   b. Select **Java EE** and click **OK**. See Figure 9-12.

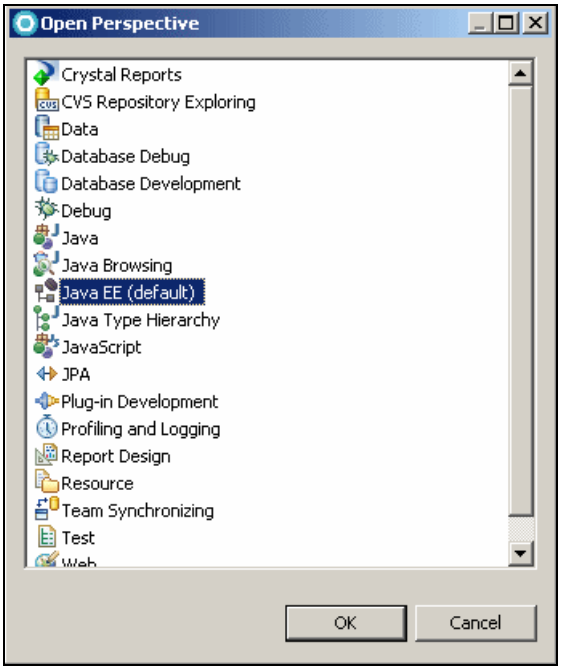


Figure 9-12 Resetting the RAD perspective to a Java EE Project look and feel

4. Create a dynamic web project by navigating to **File** → **New** → **Dynamic Web Project**. Select the name and general properties of the project, according to your environment (examples in Figure 9-6 on page 311 and Figure 9-7 on page 312.

   Examine the Context Root and EAR Project Name fields. They define properties and descriptors file because they affect the way your application is deployed on WebSphere Application Server and how it will be the web services URL. This is an important step when migrating web services, because current consumer applications for such services already know the URL for WSDL and the operations itself.

5. Expand the WAR Project created. Right-click and perform the following steps:
   a. Select **New** → **Other**.
   b. Select **Web Services** and click **OK**.

   The annotations (@WebService and @WebMethod) on the source code files make all things easier to migrate.

Now you can build a new client consumer application, applying the new reference to the WDSL provided by WebSphere or solve the common issues using one of the strategies suggested in 9.5, “Major issues when migrating web services applications” on page 322.

The client consumer application was developed in such a way that it connects to both web services. For example, both URLs for the WSDL files were referenced in the application project and both are used at the main() function for testing purposes. If you manage to start both application servers at the same time and run the application client consumer, you should see an output such as Example 9-9.

Example 9-9 Output messages after running the Client Consumer application for Calculator Web Services

```
Trying with WAS...
Multiplication Result = 12.0
Factorial Result of 3 = 6.0
**************************************
Trying with GlassFish
Multiplication Result = 12.0
Factorial Result of 3 = 6.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

9.5 Major issues when migrating web services applications

Keep in mind the URL for web services when migrating a web services application, because you might have to update references to these services (WSDL file).

You need to consider various approaches depending upon your environment:

- The first approach is to update and redeploy the application clients (the consumers for your web services, represented by number 1 in Figure 9-13 on page 323). For this approach you need to rewrite the application client and references point out the new WSDL provider. Depending on how many consumers use the service, it can take too much software development and deployment effort. If the web service is published on the Internet, this approach is not viable.

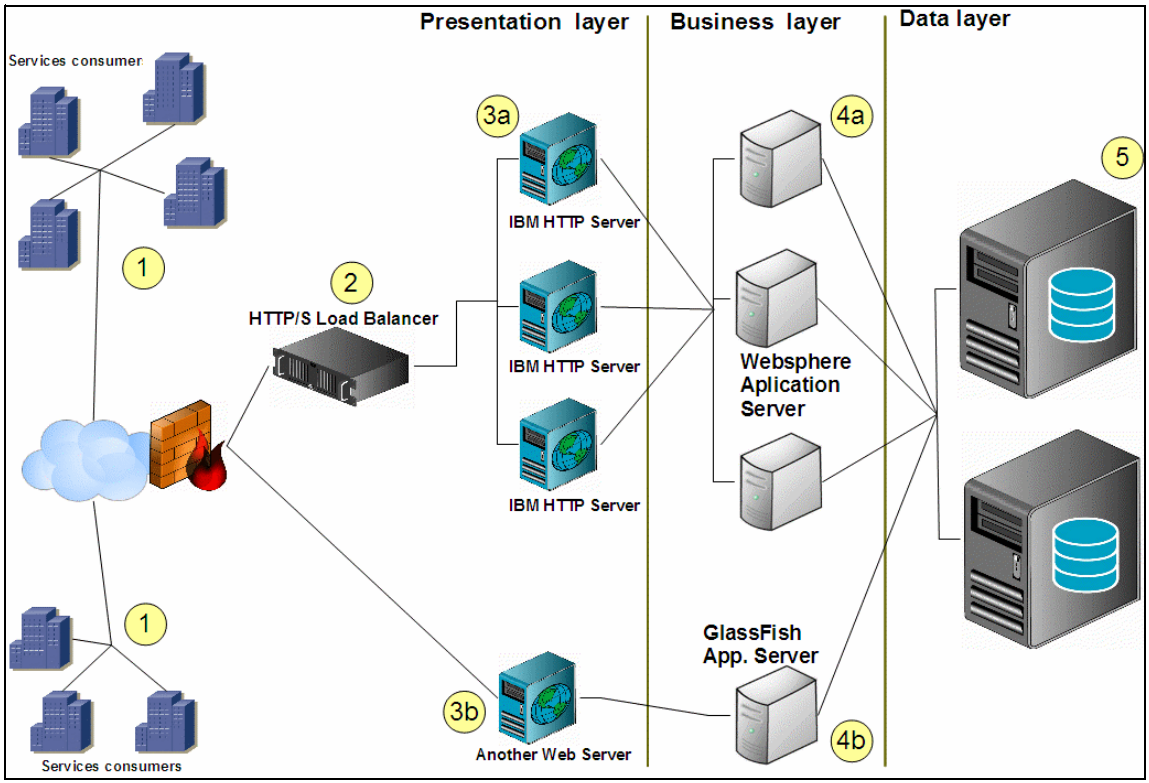


Figure 9-13 Typical infrastructure for web services applications

- If your web service was developed and installed over a three tier infrastructure, you can also make changes only on the upper tier, such as a Load Balancer, NAT, or Reverse Proxy devices (see number 2 in Figure 9-13). In this case, you need to reconfigure these devices to point to the new URL for the web service.
- Looking to the first tier in your architecture (see 3a and 3b in Figure 9-13), you probably have web servers as front-end devices for static pages, and another web application and proxy for the application server itself. For example, if you using IBM HTTP Server (IHS), you can use the HTTP modules to replace the old URLs with the new ones. For more information, visit the Apache HTTP Modules page (mod_redirect, mod_proxy, mod_rewrite, mod_alias) at the Apache HTTP web page:

  `http://httpd.apache.org/docs/2.2/mod/`

- Because you are likely moving from one application server to another, (layer 4 in Figure 9-13 on page 323) you should consider development issues when building it on RAD. If possible, try to avoid changes on the URLs (server name and server port) and path names (context root and servlet mappings). If you do not change these, it is easier to migrate and keep connectivity with other client applications. It is not possible to keep the same URLs, because you might have to move to other servers or even to another infrastructure. In these cases, you should consider the other options to remap the URLs.

9.6 Summary

As expected, we did not face a lot of problems when migrating Java EE 5 applications from GlassFish to WebSphere.

The first application, Sun Java Pet Store Reference, is only a web module in its essence (a WAR file) with Java EE features to serve as an example for practitioners and beginners to the Java EE platform. Such features include Java Server Faces (JSF) with DOJO integration, Java Persistence API (JPA), Servlets and JSPs with full usage of tag libraries. We were able to migrate this application successfully with only two changes: one for the JPA implementation and one for the compatibility issues in Web Container (URLs parser / handler).

The second application, which was developed by the authors of this book, is a simple web service application with two operations only and no additional complexities whatsoever. The web service migration is presented to raise up common issues when migrating all web services applications. Although it was originally written with Netbeans to work on GlassFish, we were able to deploy it successfully on WebSphere. As stated before, when migrating web services, there are special concerns about the client consumers for these services.

Look at 9.5, “Major issues when migrating web services applications” on page 322 for an overview of these concerns.

Because we had the source code for both applications, we used the strategy to rebuild all applications by importing the source files and external libraries to RAD. Such an approach made the migration process easy to perform and easy to deploy in WebSphere.

It is important to keep in mind that most of the GlassFish examples use only Java EE specifications with no vendor features. As usual, minor differences in JDK implementations might cause rewriting of the Java Source. Is the case of web services, it is important to observe the new environment and the new references for the web service.

A

Development practices for portable applications

The standards specified by Java Platform Enterprise Edition (Java EE) go a long way toward removing subtle distinctions between deployment platforms, making migration far easier than before. In a perfect world, Java EE applications can be moved from one application server to another one without modification. But such lack of complexity is not always present as we see in the previous chapters.

Additionally, software engineering and general software development practices, such as Design Patterns help developers (even those with no great experience) to write flexible and robust software and also help administrators to perform a better migration experience.

In this appendix, we provide a high-level overview, of the most common best practices that migration projects, or any other project, can benefit from following.

This appendix is organized in the following sections:

- Java EE development practices
- Software engineering and general development practices

© Copyright IBM Corp. 2010. All rights reserved.

Java EE development practices

In the following section we discuss the most important Java EE development practices for creating portable applications.

J2EE, standards and proprietary solutions

The evolution of standards such as Java EE and the Java language itself have done a lot to make migration a relatively easy task. That is one of the primary benefits of standards. But standards take time to evolve. In the meantime, solutions are required for problems that grow at an fast rate. In the early days of server-side Java, for example, a number of HTML/Java hybrids were created to solve a problem that was eventually solved by the JavaServer Pages (JSP) standard.

Java EE packages together a large set of specifications that provide a set of services for server-side application development. A Java EE-compliant server provides almost everything you need to build an enterprise application. As comprehensive as Java EE is, however, it does not cover everything that you can ever possibly care about. In the evolving world of Java development, it is impossible for any one set of standards to cover everything. Needs change too quickly. For example, portals and personalization are not currently governed by standards but are an important part of many applications. Proprietary solutions are developed to fill gaps in the standards. Sometimes these proprietary solutions evolve into standards. More often, complementary standards are developed.

You might have chosen your application server, database, middleware, or whatever, based on proprietary services that are provided. You might be moving to IBM WebSphere to take advantage of its proprietary services. These services are often what distinguish competitive products from one another. Proprietary solutions are not inherently bad, but can tie you to a specific vendor and make future migration and change dependent on that vendor's continued support of those solutions.

Even services covered by standards are not completely safe. As part of Java EE, Java Database Connectivity (JDBC) provides a standard mechanism for accessing relational databases from Java. JDBC encourages the creation of standard Java code, but does little to stop you from taking advantage of specific database extensions. The JDBC standard is only effective at insulating you from the specifics of the actual database if you use it with care to avoid proprietary extensions.

Conform to standards as much as possible

Standards can be tricky because there are so many to choose from. Standards are only as solid as the community that supports them. Java EE, from all appearances, is a good standard in which to put faith. However, it is important to approach the use of standards with care. As mentioned previously, even standards such as JDBC can be troublesome.

Perhaps more important than just conforming to standards, is adopting a philosophy of keeping pace with standards. Standards often go through multiple versions, which can make conformance more difficult. As Java EE evolves, new features are added and older ones are retired. The retirement cycle is usually lengthy, providing lots of time for application developers to catch up. In Java parlance, types and methods that are due to retire are marked in advocate as deprecated. Development tools, such as Rational Application Developer for WebSphere “(see 3.3.1, “Rational Application Developer for Assembly and Deploy V7.5” on page 64) help you keep up with standards by identifying the use of deprecated code.

Make conformance to standards a high priority in your development plan. Not only for initial development, but as a long-term development plan. As the standards are updated, work time into your development plan to update your code to maintain conformance to those standards.

Abstraction is a key point

Abstractions are a way of putting a single, consistent face on functionality. Rather than letting every part of your application make direct use of technology, make those parts access the technology through an abstract layer. If you later need to change the technology in question, you only have to change the code under the abstraction and everything that makes use of the abstraction should work.

Develop the application architecture into layers

Layering is a way of grouping an application's code into related subparts. Each layer contains code that is specific to a certain function or purpose. A typical example of layering is the so-called Model/View/Controller (MVC) architecture, which separates model, view, and control into layers.

Layers provide many advantages. The most obvious is separation of roles and responsibility. Content designers are responsible for JSPs, Java developers are responsible for servlets, JavaBeans, and EJBs. It goes deeper than that, however. As change is introduced into an application (migration is just one example of change), layering helps by isolating entire chunks of code into easier

to understand bits. Further, these bits are easy to test. When layering is done well, the entire business logic of an application can often be migrated with little or no modification.

Write and maintain automated tests

Tests are the only way you know that your migration works. Manual testing can take a long time and is prone to errors, though it does have its place. Automated tests are a valuable mechanism for software developers to validate their work on a frequent basis.

Automated testing can take many forms. The JUnit testing framework is a simple Java-based testing framework that is available at no charge (under open-source licensing). Test-infected JUnit users develop tests before developing the code being tested. To the uninitiated, this sounds strange. But it works. With proper discipline, you are guaranteed to have test code in place.

Note: For more information about JUnit, visit the following web page:

`http://www.junit.org`

One of the advantages of the MVC architecture (and the Java EE implementation of MVC) is that the componentization of the elements make it possible (in fact, relatively easy) to test your application in pieces. Therefore, you can easily write tests to test separately persistence, session beans, and portions of the user interface outside of the rest of the code base. There are a number of frameworks and tools for Java EE testing that make this process easier. For instance, JUnit, which is an open-source tool developed by junit.org, and Cactus, which is an open source project of the Apache consortium, are both quite useful for testing Java EE components.

Despite all of the great information about deeply testing your application, we still see many projects where it is believed that if the GUI is tested (which might be a web-based GUI or a stand-alone Java application), then the entire application has been comprehensively tested. GUI testing is rarely enough. There are several reasons for this.

- With GUI testing, it is difficult to test every path through the system. The GUI is only one way of affecting the system. There might be background jobs, scripts, and various other access points that also need to be tested. These often do not have GUIs associated with them.

- Testing at the GUI level is coarse-grained. A GUI tests how the system behaves at the macro level of the system, meaning that if problems are found, entire subsystems must be considered, making finding any bugs identified extremely difficult.
- GUI testing usually cannot be done well until late in the development cycle when the GUI is fully defined. This means that latent bugs will not be found systematically until late.
- Average developers probably do not have access to automatic GUI testing tools. Thus, when a developer makes a change, there is no easy way for that developer to retest the affected subsystem. This actually discourages good testing. If the developer has access to automated code level unit tests, the developer can easily run them to make sure the changes do not break existing function.
- If automated builds are done, it is fairly easy to add an automated unit testing suite to the automated build process. By doing this, the system can be rebuilt regularly (often nightly) and regression tested with little human intervention.

In summary, by using automated unit tests, defects are found sooner, defects are easier to find, testing can be made more systematic, and thus, overall quality is improved.

Documentation

Undocumented code is next to useless. Documentation best practice is fundamental to the design, implementation, and maintenance phases. If code is not properly documented, code maintenance in subsequent iterations can be at best problematic, and at worst, impractical. Code maintenance has many sub areas. Extending, scaling, and fixing applications are all forms of maintenance. Without good documentation, these tasks are much harder to complete.

The following list details best practices for creating good documentation for your application code:

- JavaDoc gives developers a contract

  JavaDoc is used to document classes, instance variables, and methods. It has clearly defined annotations that can aid the development process considerably. For example, the method JavaDoc comments have annotations to describe input parameter values, output values such as return values and exceptions, and under what circumstances these are produced. As such, the JavaDoc comment method, if properly implemented, represents the method contract.

Given that JavaDoc gives such a precise way of documenting the modules in an application, JavaDoc documentation of code is not something that is left to chance. Indeed, JavaDoc comments are artifacts in the development process that are written at design time. There are various ways of doing this.

A low-level design document might describe all the modules that are to implement the system. The JavaDoc comments can then be written into the low-level design document. The low-level design then forms the template for developers who have been allocated particular modules. They can cut and paste the JavaDoc comments into their source files.

A design tool such as Rational Rose® can be used to design components. As components are created, they can have their JavaDoc class, instance variable, and method comments filled in. When the design is complete, design documentation can be generated in the form of UML diagrams through automated design documentation tools such as Rational SoDA®. Additionally, Java class shells can be generated. That is, Java source code can be generated from the Rational Rose model including all the class, instance variables, and method signatures along with all the JavaDoc comments. These classes can then be allocated to each developer for implementation.

Whether using design documents with JavaDoc comments in them or more sophisticated design tools such as Rational Rose that generate code shells, the outcome of this approach is that the JavaDoc comments that form contracts are placed in front of the developer before they start coding. This keeps the contract for the module in front of the developer as he codes the module. By sticking to the class and method contracts the developer is more likely to achieve the desired outcome: a module that is going to pass unit test first time around.

- Local comments best practice: why, not what

  During coding, code is commented where necessary. The comments we are talking about in this case are not JavaDoc comments. They are local comments written in an algorithm inside a method body. The general best practice rule of thumb for writing effective comments is that the comments should explain 'Why?' and not 'What?'. Well-designed, well-written code is itself a documentation of what is happening. If code is so poorly designed or esoterically written that what is happening is not clear, the developer should consider being more explicit in their coding style or re-factoring the code. Why something is being done is not always so clear locally. Variables and method calls often come from separate modules. Non-procedural languages do not put everything in front of the developer at the same time. Explaining why something is being done in a particular way is what local comments are used for.

- Saving time in code reviews

  Good documentation of code saves time during code reviews. It allows other developers to read code easily. It also cuts down on findings from reviews. One of the major, and unnecessary, types of finding in code review is the need for more accurate JavaDoc and local comment blocks, if those blocks are found

Use MVC

Cleanly separate business logic (Java beans and EJB components) from controller logic (servlets/Struts actions) from presentation (JSP, XML/XSLT). Good layering can cover a multitude of sins.

This practice is so central to the successful adoption of Java EE that there is no competition for the #1 slot. Model-View-Controller (MVC) is fundamental to the design of good Java EE applications. It is the division of labor of your programs into the following parts:

- Those responsible for business logic (Model: Often implemented using EJBs or plain old Java objects)
- Those responsible for presentation of the user interface (View).
- Those responsible for application navigation (Controller: Usually implemented with Java servlets or associated classes such as struts controllers).

There are a number of problems that can emerge from not following basic MVC architecture. Most problems occur from putting too much into the View portion of the architecture. Practices such as using JSP tag libraries to perform database access, or performing application flow control within a JSP are relatively common in small-scale applications, but these can cause issues in later development as JSPs become progressively more difficult to maintain and debug.

Figure A-1 shows a typical Java EE MVC implementation.

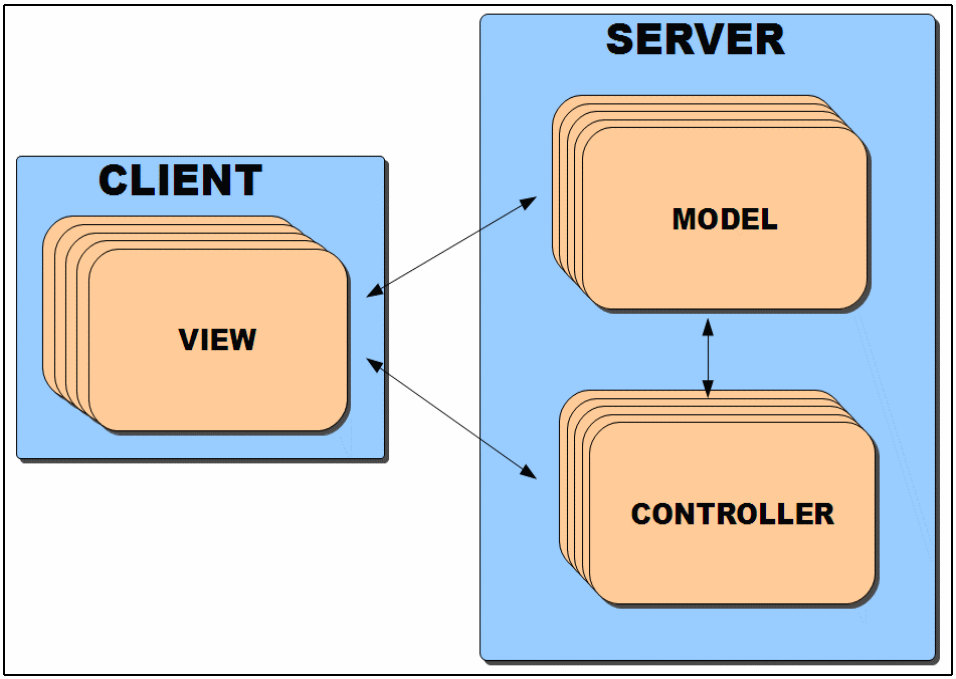


Figure A-1 Typical Java EE MVC implementation

Likewise, we often see migration of View layer constructs into business logic. For instance, a common problem is to push XML parsing technologies used in the construction of views into the business layer. The business layer should operate on business objects, not on a particular data representation tied to the view.

However, just having the proper components does not make your application properly layered. It is quite common to find applications that have all three servlets, JSPs, and EJB components, where the majority of the business logic is done in the servlet layer, or where application navigation is handled in the JSP. You must be rigorous about code review and re-factoring to ensure that business logic is handled in the Model layer only, that application navigation is solely the province of the Controller layer, and that your Views are concerned with rendering model objects into appropriate HTML and Javascript.

User interface technologies change rapidly and tying business logic to the user interface makes changes to just the interface deeply impact existing systems. Just a few years ago, user interface developers for web applications can choose from servlets and JSPs, struts, and perhaps XML/XSL transformation. Since then, Tiles and Faces have become popular, and now AJAX is gaining a strong following. It is a shame to have to redevelop an application's core business logic every time the preferred user interface technology changes.

Struts

Struts is a framework that is implemented according to the MVC pattern. Prior to Struts, developers used combinations of JSPs and Servlets to create the View and Controller layers of an application. Struts took the best practices that emerged from the JSP and Servlet architectures and created an architecture that introduces an action model where JSP form data is submitted to an ActionServlet controller.

Struts uses JSP tag libraries extensively. WebSphere Application Server V6 now comes with caching features specific to Struts architectures to reduce the overhead of tag library references.

Where possible, JSP, rather than XSLT, is used to render the view layer as XSLT (which is the transformation of XML to HTML pages) is expensive. Form beans should not be passed to the business logic layer. The layers are completely separate. Passing form beans to the back end means that the back end is not independent from the front end in terms of the objects it needs to know about.

De facto standard for MVC frameworks: Struts is the de facto standard for MVC frameworks, but there are many other open source frameworks that are more lightweight and easier to use. Java Server Faces (JSF) is a standardization effort that tries to unify a fragmented market.

For more information about Apache Struts Project, visit the following web page:

`http://struts.apache.org/`

Java Server Faces (JSF)

Java Server Faces is a specification, defined by the JSR-252 Java Community Process, which aims at simplifying the creation of complex GUIs in web applications. JSF is similar to Swing and other client APIs in that it provides a set of standard widgets and a framework for creating additional widgets.

Java Server Faces splits the user interfaces presentation and processing cleanly, which allows for easier construction of multiple user interfaces. The components available are varied and sophisticated, which increases the possibility of what GUI components can be used in the front end. This richness of features and the further level of indirection between the interface and processing comes at a cost. JSF can be relatively poor in performance when compared with JSP and Struts. The decision to use the technology has to weigh the merits of a sophisticated and extensible interface against the performance costs.

Do not reinvent the wheel

Use common, proven frameworks such as Apache Struts, JavaServer Faces, and Eclipse RCP. Use proven patterns.

Such frameworks are not only well-accepted in the Java community, but fully supported within the WebSphere runtimes and Rational tool suites as well. Likewise, in the rich client arena, the Eclipse RCP (Rich Client Platform) has also gained wide acceptance for building stand-alone rich clients. While not a part of the Java EE standard, these frameworks are now a part of the Java EE community, and are accepted as such.

But it is still common to see many companies maintaining or even developing new user-interface frameworks that are functionally equivalent to Struts or JSF. There are many reasons why this can be true:

- Organizational inertia,
- “Not invented here” syndrome,
- Lack of perceived benefit in changing working code,
- A slight sense of hubris in thinking that you can do things better than the open-source developers did in a particular framework.

Develop to the specifications, not the application server

Know the specifications by heart and deviate from them only after careful consideration. Just because you can do something does not mean you should.

There are several places in which not taking the most straightforward approach can cause problems. A common one today is where developers take over Java EE security through the use of JAAS modules rather than relying on built-in specification-compliant application server mechanisms for authentication and authorization. Be wary of going beyond the authentication mechanisms provided by the Java EE specification. This can be a major source of security holes and vendor compatibility problems. Likewise, rely on the authorization mechanisms provided by the servlet and EJB specs. When you need to go beyond them, make sure you use the specifications's APIs (such as getCallerPrincipal()) as the basis for your implementation. This way, you are able to make use of the vendor-provided strong security infrastructure and, where business needs require, support more complex authorization rules.

Other common problems include using persistence mechanisms that are not tied into the Java EE specification (making transaction management difficult), relying on inappropriate Java Standard Edition facilities (like threading or singletons) within your Java EE programs, and individualized solutions for

program-to-program communication instead of staying within supported mechanisms such as Java 2 Connectors, JMS, or web services. Such design choices cause no end of difficulty when moving from one Java EE-compliant server to another, or even when moving to new versions of the same server. Using elements outside of Java EE often causes subtle portability problems. The only time you should ever deviate from a specification is when there is a clear problem that cannot be addressed within the specification. For instance, scheduling the execution of timed business logic was a problem prior to the introduction of EJB 2.1. Today, of course, the EJB specification now provides for time-based function so we encourage the use of standard interfaces. In this way, maintenance and migration to later specification versions becomes the problem of the vendor, and not your problem.

Build what you know

Iterative development enables you to master all the moving pieces of Java EE gradually. Build small, vertical slices through your application rather than doing everything at once.

Due to the size of Java EE standards, it is a better approach to take one step at a time. If a development team is just starting with Java EE, it is far too difficult to try learning it all at once. There are too many concepts and APIs to master. The key to success in this environment is to take on Java EE in small, controlled steps.

This approach is best implemented through building small, vertical slices through your application. After a team has built its confidence by building a simple domain model and back-end persistence mechanism (perhaps using JDBC), and has thoroughly tested that model, they can then move to mastering front-end development with servlets and JSPs that use that domain model. If a development team finds a need for EJBs, they can start with simple session facades atop container-managed persistence EJBs or JDBC-based DAOs (Data Access Objects) before moving on to more sophisticated constructs, such as message-driven beans and JMS.

Consider adopting the new Agile development methods, such as Extreme Programming (XP), that foster this kind of incremental learning and development. There is a procedure often used in XP called *ModelFirst* that involves building the domain model first as a mechanism for organizing and implementing your user stories. You build the domain model as part of the first set of user stories you implement, and then build a user interface on top of it as a result of implementing later user stories. This fits with letting a team learn technologies one at a time, as opposed to sending them to a dozen simultaneous classes (or letting them read a dozen books), which can be overwhelming.

Iterative development of each application layer fosters the application of appropriate patterns and best practices. If you begin with the lower layers of your application and apply patterns such as Data Access Objects and session facades, you should not end up with domain logic in your JSPs and other View objects.

Always use session facades whenever you use EJB components

Use local EJBs when architecturally appropriate.

Using a session facade is a well-established best practice for the use of EJBs. In fact, the general practice is widely advocated for any distributed technology, including CORBA, EJB, and DCOM. The lower the distribution cross-section of your application, the less time is wasted in overhead caused by multiple, repeated network hops for small pieces of data. The way to accomplish this is to create large-grained facade objects that wrap logical subsystems and that can accomplish useful business functions in a single method call. Not only does this reduce network overhead, but within EJBs, it also critically reduces the number of database calls by creating a single transaction context for the entire business function. This is actually one of the core principles of Service Oriented Architecture (SOA).

Use stateless session beans instead of stateful session beans

This makes your system more amenable to failover. Use the HttpSession to store user-specific state.

Java EE application servers providing for stateful session bean failover can work around issues, but stateful solutions are not as scalable as stateless ones. For example, in WebSphere Application Server, requests for stateless session beans are load-balanced across all of the members of a cluster where a stateless session bean has been deployed. In contrast, Java EE application servers cannot load-balance requests to stateful beans. This means load might be spread disproportionately across the servers in your cluster. In addition, the use of stateful session beans enforces state information to your application server, which is undesirable. Stateful session beans increase system complexity and complicate failure scenarios. One of the key principles of robust distributed systems is the use of stateless behavior whenever possible.

Use container-managed transactions

Learn how two-phase commit transactions work in Java EE and rely on them rather than developing your own transaction management. The container is almost always better at transaction optimization.

Using container-managed transactions (CMTs) provides two key advantages that are nearly impossible to obtain without container support: composable units of work, and robust transactional behavior.

If your application needs to access multiple resources as part of the same operation, you need two-phase commit transactions. For example, if a message is removed from a JMS queue and a record is updated in a database based on that message, it is important that either both operations occur or that neither occurs. If the message was removed from the queue and the system failed without updating the database, this system is inconsistent. Serious customer and business implications result from inconsistent states

Prefer JSPs as your first choice of presentation technology

Use XML/XSLT only in cases where you have multiple presentation output types that must be supported by a single controller and back-end.

Choose JSP as your first option for presentation technology because it is the best-supported and best-understood Java EE view technology available. Given the introduction of custom tag libraries, the JSTL, and the JSP 2.0 features, it has become increasingly easy to build JSPs that do not require any Java code, and that cleanly separate Model and View. There is significant support (including debugging support) for JSP built into development environments, such as IBM Rational Application Developer. Many developers find developing with JSP easier than developing with XSL due to how JSP is procedure-based, as opposed to rules-based. While Rational Application Developer supports XSL development, the graphical layout tools and other features supporting JSP (especially when in the context of frameworks such as JSF) make it much easier for developers to work in a WYSIWYG way, which is something that is not easily done with XSL.

Enable session persistence

When using HttpSessions, store only as much state as you need for the current business transaction and no more.

HttpSessions are great for storing information about application state. The API is easy to use and understand. Unfortunately, developers often lose sight of the intent of the HttpSession, which is to maintain temporary user state. It is not an arbitrary data cache.

One common problem is in using HttpSessions to cache information that can be easily recreated, if necessary. Because sessions are persisted, this is a expensive decision forcing unnecessary serialization and writing of the data. Instead, use an in-memory hash table to cache the data and keep a key to the data in the session. This enables the data to be recreated if the user fails over to another application server.

If you do not enable session persistence, and a server is stopped for any reason (a server failure or ordinary maintenance), any user that is currently on that application server loses their session. That makes for a unpleasant experience. They have to log in again and redo whatever they were working on. If session persistence is enabled, WebSphere automatically moves the user (and their session) to another application server, transparently. They will not even know it happened.

Take advantage of application server features that do not require your code to be modified

With features such as WebSphere Application Server caching and the Prepared Statement cache, the performance gains are substantial and the overhead is minimal.

Be prudent in applying application-server-specific features that modify your code. It makes portability difficult and might make version migration challenging as well. However, there are a suite of application-server specific features available, particularly in WebSphere Application Server, that do not modify your code. Your code is written to the specification, but if you know about these features and how to use them properly you can take advantage of significant performance gains.

For example, in WebSphere Application Server you should turn on dynamic caching and use servlet caching. The performance gains are substantial and the overhead minimal, and the programming model is unaffected. The merits of caching to improve performance are well understood. Unfortunately, the current Java EE specification does not include a mechanism for servlet/JSP caching.

However, WebSphere Application Server provides support for page and fragment caching through its dynamic cache function without requiring any application changes. The cache policy is specified declaratively and configuration is through XML deployment descriptors. Therefore, your application is unaffected, remaining Java EE specification compliant and portable, and benefiting from the performance optimizations provided from WebSphere's servlet and JSP caching. The performance gains from dynamic caching of servlets and JSPs can be substantial, depending on the application characteristics

Play nice within existing environments

Deliver a Java EE EAR and configurable installation scripts, not a black box binary installer.

In most realistic scenarios, large WebSphere Application Server users run multiple applications in the same shared cell. This means that if you provide an application to be installed, it must install reasonably into an existing infrastructure. This means two things:

- You must limit the number of assumptions you make about the environment.
- You cannot anticipate every variant, so the installation process must be visible.

Deliver an EAR file (or a set of EAR files), along with documentation and installation scripts. The scripts should be readable so that the installer can understand what they do and can validate that there is nothing dangerous being done by the scripts. For situations where the scripts are inappropriate, users might need to install your EARs using other processes that they already use, which means you must document what your installer is doing.

Embrace Java EE, do not fake it

Commit to building real Java EE applications that truly make Use of Java EE function.

It is common to see applications that claim to run in WebSphere but are not really WebSphere applications. There are several examples where there is a thin piece of code (perhaps a servlet) in WebSphere Application Server and all of the remaining application logic is actually in a separate process. For example, a daemon process written in Java, C, C++, or whatever, but not using Java EE, does the real work. That is not a real WebSphere Application Server application. Virtually all of the qualities of service that WebSphere Application Server provides are not available to such applications. This can be quite a rude awakening for users that think they are using an Java EE application.

Plan for version updates

Change is inevitable. Plan for new releases and fix updates so that your customers can stay current.

WebSphere Application Server continues to evolve, and so it should not surprise you that IBM regularly produces fixes for WebSphere Application Server, and that IBM periodically releases new major versions. You need to plan for this. There are two kinds of development organizations that this impacts: in-house developers and third party application vendors. The basic issues are the same, but each is impacted differently.

Log your program state using a standard logging framework

This includes exception handlers. Use a logging framework such As JDK 1.4 logging or Log4J.

Logging is sometimes the most tedious, undervalued part of programming, but it is the difference between long hours of debugging and going home at a reasonable time. As a general rule of thumb, at every transition point, log it. When you are passing parameters from one method to another method, or between classes, log it. When doing a transformation on an object, log it. When in doubt, log it.

Once you have made the decision to log, choose an appropriate framework. There are lots of good choices out there but we are partial to the JDK 1.4 trace APIs, as they are fully integrated into the WebSphere Application Server trace subsystem and are standards-based.

Always clean up after yourself

If you obtain an object from a pool, always make sure you return it back to the pool.

One of the most common errors we see with Java EE applications, whether running in development, test, or production, are memory leaks. Nine times out of ten, it is because a developer forgot to close a connection (JDBC most of the time) or return an object back into the pool. Make sure that any objects that need to be explicitly closed or returned to the pool are so done. Do not be one of the culprits responsible for the offending code.

Follow rigorous procedures for development and testing

This includes adopting and following a software development methodology.

Large scale system development is hard and it should be taken seriously.

Almost any method works for most teams provided that they are well-understood by the team members, followed rigorously, and adjusted carefully to deal with the specific natures of the technology and team that is using that method.

Software Engineering and general development practices

Any development effort benefits from the best practices discussed in the following sections.

Emphasis is placed on an iterative and object oriented approach because these are the prevailing trends in contemporary project management and development

Design patterns

If you have a problem to solve, you can trust that it has been faced and solved before. Commonly used solutions in application architectures fall into categories. These categories are called *design patterns*. The following subsections discuss generic design patterns that are not specific to Java EE or to the Java language, but are extensively used in Java EE.

How to get more information about Java EE design patterns: For further information about Java EE design patterns, visit the following web pages:

`http://java.sun.com/blueprints/patterns/`
`http://www.javacamp.org/designPattern/`

Important: It is important to know when to use a pattern and how to apply it correctly. This knowledge can only be gained through experience and practice. Java and J2EE applications tend to overuse patterns, which leads to unnecessary complexity.

Data Access Objects (DAO)

Applications that store and retrieve data from a persistent storage, such as a database, benefit from using the Data Access Objects (DAO) pattern. The DAO pattern is similar to a facade object that provides a simple interface to client modules in that it implements the connection and retrieval code internally.

The DAO pattern hides the implementation details from the client by encapsulating details related to data access. This makes it easier to change the type of persistent storage (for example from an RDBMS to XML files) and the data access mechanism (for example switching from entity EJBs to Hibernate). Using EJBs for accessing databases and other resources is not always necessary, and if this is the case, the memory and network overheads of using an EJB should be avoided.

Service Locator

J2EE application servers expose services and components like EJBs through JNDI. A single object can be used to do JNDI lookups for resources such as EJBs and Java messaging resources. Delegating this task to one object that is then called by other objects is far better practice than all objects doing the same JNDI call to look up objects.

EJBHomeFactory

This is a composite design pattern that uses other design patterns. Its purpose is to reduce the overheads when looking up home interface references for EJBs. This design pattern is similar to the service locator pattern. It also uses the singleton pattern, which means it maintains a single instance of itself for use by other modules. It also uses the factory design pattern (that is, its sole purpose in life is to produce instances of a particular object type for other modules).

Facades

The facade design pattern offers a simplified interface by encapsulating a complex interface or collection of interfaces. A session facade pattern is constructed from a stateless session EJB accessing, for example, one or more Entity EJBs. The session facade is based on the facade design pattern. A facade simplifies the interface of a more complex underlying system. In a session facade, the complexity of finding and calling a number of entity beans is simplified to one call to a method on a Session bean. This pattern, in turn, reduces complexity in client layers.

This pattern reduces network overhead, caused by remote method invocations, by encapsulating multiple remote calls into one.

This pattern also reduces transaction handling overheads by not allowing view objects to access entity beans directly. When view objects access entity beans directly, separate transactions are created which is an extra overhead.

Singletons

Singletons are classes that return an instance of themselves to calling classes. They only return a new instance of themselves to calling classes, if the single instance of themselves that they maintain has not been created yet. The advantage of this is the avoidance of static method calls. Static method calls are not a clean OO approach. Moreover, ensuring that only one instance of a class exists at any one time means all users of the object reference that object rather than maintaining their own instance. Separate instances represent a memory overhead.

However, a word of caution about singletons: singletons are only desirable if the object is totally generic in all situations for all calling classes. If this is not the case, the singleton pattern should not be selected.

Factories

The factory design pattern is where an object has a dedicated purpose. The purpose is to produce instances of a particular object class. The factory object is the single point of access for obtaining an instance of the object. The factory design pattern is often used in conjunction with the singleton pattern that ensures that only one instance of the factory object exists in memory.

Dependency Injection (DI)

Dependency Injection, also known as Inversion of Control, is a pattern that delegates the process of instantiating objects and setting up the dependencies between these objects to a light-weight container. The Dependency Injection pattern provides the following benefits:

- Removes hard coded dependencies between classes

  The container takes care of the dependencies and the creation of objects.

  Dependency Injection promotes the principle of coding to interfaces instead of concrete classes. Using interfaces allows you to change the implementation with just a configuration change.

- Configuration flexibility

  Configuration of components is done in one central place, the container configuration. This makes it easy to change application details without recompiling source code.

- Improves testability

  JUnit test cases usually do not depend on external systems such as databases or a J2EE container. By coding to interfaces, you automatically improve testability. Changing the concrete implementation of an interface (for example changing a DAO to use an in-memory database instead of an RDBMS) is done in one place only, the container configuration.

There are several implementations of the Dependency Injection pattern that are available for download, such as:

- PicoContainer

  PicoContainer is a light-weight container that implements the Dependency Injection pattern. For more information, see the following web page:

  http://www.picocontainer.org

- Spring Framework

  Spring Framework contains an implementation of the Dependency Injection pattern. Spring Framework is a more general J2EE framework, and includes many other features than just Dependency Injection. For more information, see the following web page:

  http://www.springframework.org

Tip: Dependency Injection removes the need for implementing many patterns, including the factory, singleton, and service locator patterns. Dependency Injection is a better solution than these patterns combined because it allows you to create code that is less tightly coupled.

Iterative development

During development, there are always unknowns. These unknowns can take the form of the use of new technology, the use of new development staff, or the use of new ideas to solve problems. These unknowns then take the form of risks. The earlier these risks are exposed, the earlier they can be mitigated.

Developing iteratively means to go through a number of sub-development cycles in turn to expose risks early. This is in contrast to the waterfall model. In a waterfall development model, because testing occurs late in the cycle, risks (such as the use of new technology and its unknowns) are not exposed until the components that use that technology are built. Worse, they might not be exposed until those components are unit-, integration-, and acceptance-tested.

Iterative development steps through sub-iterations of the entire development process. This means that fundamental parts of the application are prescribed, designed, implemented, tested, and deployed in controlled iterative cycles. The key word here is *controlled*. The objective of the iteration has to be clear and self-contained. Discipline in project management and software development throughout the micro-iteration has to be as tight as it is in the entire macro-iteration.

For example, a project called *Redbook Application* might require the creation of a web front end using Struts technology, an EJB back end including the use of Java Messaging Service technology and an EJB client application. The front and back layers might require many interface and back-end components. However, a first iteration might design, code, and test a single web interface component, a single back-end component, and a single EJB client component. The web interface makes use of Struts in this first iteration and the back-end component makes use of Java Messaging Service technology at this early point.

This first iteration is not implementing all the web interface, backed EJB, and EJB clients needed. But what it is doing is creating a single example of each. It is also creating parts of the application that are needed rather than prototyping with functionally irrelevant example code. Finally, it is implementing and testing the use of the specific technologies (such as Struts and Java Messaging Service), which might be new to members of the development team.

Furthermore, there might be novel ideas or new design patterns that are being used. An example of these are included also in this first iteration. It is important note that the iteration is producing functionality that is needed in the final application. The iterative approach is similar to prototyping. Prototyping is the practice of building an application to prove theories from the design stage. It is also possible to show a prototype to users and customers to ask “Is this the sort of thing you wanted?” It is commonly a stage somewhere between requirements gathering, design, and coding stages in the waterfall model.

The danger of prototyping are that when it produces an application that tests theoretical and new unknown areas, it is not focused on necessarily producing functionality required in the end product. It is often discarded, which can be seen as a waste of development effort. On the other hand, and in the worst case, if the prototype is concerned with producing final functionality, it might do so in a way that is not robust or well designed, just to get something up and running. This poor design and lack of robustness can then find its way into the final product. It has happened that clients, when shown a prototype, think it is so good (and cheap, because they have something that works in a short period of time) that they want to use that as the final product.

The iterative approach takes the benefits of prototyping but avoids its pitfalls. The deliverables of the iteration are deliverables for the final product. By following the iterative approach, risks are identified early and mitigated.

Absorbing changing requirements

The iterative approach also allows changing requirements to be absorbed. Requirements change. Life is not static and even during a short development project, there is enough time for customers or users to change their mind. These changes might have good reasons behind them. The environment in which the application is to be used might change. The market to which the business is responding might change. The experience of the users or domain experts might change (often as a result of focusing on the requirements for the application). These changes are not necessarily to be discouraged, as they can lead to genuine and necessary improvements in the original requirements and designs. If an iterative approach is being followed, these changing requirements can be absorbed more easily during the current iteration.

Quality

Quality improves through iterative development. By definition, the application goes through a number of design, coding, and testing cycle revisions. This enhances the application's robustness, performance, and maintainability.

Learning

Members of the team learn lessons. They can apply the knowledge in the same macro-development cycle or project. If a team member needs specific training, these needs are identified earlier and the knowledge learned can be applied in the same project. Team members also learn about the development process itself. The process or the team member's practice of the process can be tweaked and improved upon.

Reuse

Reusable components, or common components, might turn out to be not so generic in practice. Likewise, components, which were thought not to be generic, might emerge as generic as other uses become apparent. These design considerations might not emerge until the project is under way. Also, the use of third party technology (for example a GNU API such as Apache Xerces XML API) might be evaluated. Sometimes, third party technology can turn out to be inappropriate and another alternative needs to be sought; sometimes the technology might have additional, unexpected benefits that one likes to take advantage of too. The iterative approach makes these adjustments possible.

In summary, software development projects are learning processes no matter how well planned they are. The iterative model allows for controlled and sub-iterations of the entire life cycle of the software development. This iterative development model allows the team to take advantage of new knowledge earlier. Hindsight is taken advantage of before the macro-iteration of the project life cycle ends.

Requirements definition

Capturing requirements and managing changes to those requirements is fundamental to a successful project. Clear requirements and a process for managing changes to those requirements gives developers a chance to design and implement the right system. If requirements are vague and the process for managing changes to requirements is poor or non-existent, the deliverables are going to be late and might be late and incorrect.

The employment of use cases is a best practice when modeling requirements. These use cases can be used to design components and devise test cases. A use case is a scenario in the application that has actors and outcomes. It is a model of the permutations of events that might occur or be allowed to occur during that scenario. Use case diagrams are a part of the Unified Modeling Language (UML) and are a standard way of expressing these scenarios. It is possible to write use cases without creating use case diagrams. It is highly recommended that use cases be identified and created during the requirements stage to achieve a successful development project outcome.

Object Oriented (OO) approach to design and programming

This seems an obvious statement, given the use of contemporary OO languages, but it is an important point to make because, as is commonly said, it is all too possible, but undesirable, to use OO languages in a procedural way (just as it is possible to write OO programs using procedural languages).

Developers using OO languages do not necessarily follow OO best practices or think in a component-based manner. Developers must think in a modular way during design, coding, and testing. The application is a module made up of smaller components working together. The key concepts considered are *cohesion* and *coupling*. Developers should aim for high cohesion and low coupling in modules.

High cohesion means that areas of common data and behavior are encapsulated in a single object or related set of objects. This approach groups behavior and data into manageable and reusable components.

Low coupling means that the interfaces of each component that allow the components to interact is kept as simple as possible and calls to interfaces are kept to a minimum.

Tip: The success of the approaches is cumulative. If the right decisions are made in making modules and groups of modules cohesive, the goal of low coupling is easier to achieve.

If you do not understand this concept, consider the placing of a fire hose on an ambulance rather than on a fire truck. No doubt the ambulance is sometimes in the same place as the fire truck and the firefighters can use the hose. But clearly, it is better if the hose were part of the fire truck. This might seem like common sense but in the abstract world of software development, such clarity can sometimes be harder to come by. When in doubt, applying first principles (such as the laws of high cohesion and low coupling) can help clarify the situation.

Good OO approaches and thinking lead to elegance, reusability, and easy maintenance. Good OO approaches and thinking are surprisingly rare among developers using OO languages and technologies. Use of design patterns and IDE tools can help guide the developer into good OO practices but understanding OO design principles is fundamental.

Tip: There are two basic OO principles that we suggest because of the many benefits that are associated with their use:

- Coding to interfaces

  This principle is coding to interfaces instead of concrete implementations. The interface is the contract between the provider of a service and the client. Changing the implementation does not break this contract; only a change in the interface can do this.

- Object composition

  This principle can be seen as an extension of the first one. Using inheritance instead of object composition causes problems because subclasses directly depend on the parent class. A change in the parent class might break classes down further in the inheritance hierarchy.

  A better strategy is to use object composition in combination with the principle of coding to interfaces. This results in finer-grained objects that are used through a facade that hides the implementation details. The objects are assembled at run time, by Dependency Injection or code, which has the added benefit of allowing you to switch the implementation.

Modeling languages

Unified Modeling Language (UML) is a visual way of expressing concepts in software development. Use cases and class architectures, for example, can be expressed unambiguously. UML does not have to be used, but developers need to have a way of communicating design ideas clearly and without extraneous details. UML class diagrams, for example, are one view of the OO component architecture. They show the entities that the developer wants to show and do not show those that confuse issues. Rational Rose and Rational Application Developer V6 both have UML diagram capabilities. It is suggested that developers use this functionality.

Note: UML can be used passively, especially with class diagrams. This is important because developers new to UML will find it far easier to read diagrams than to create diagrams. Components such as classes and packages can be created in Rational Application Developer V6 (and other Rational Tools such as Rational Rose). These components can be dropped onto diagram panes and the class diagram UML is automatically generated.

These diagrams can be exported to design documents. The auto-generation of UML means that even developers that are relatively new to UML can use it. Developers can passively read the diagrams rather than actively produce the diagrams.

Regular reviews and check points

During development, design and code reviews are invaluable. Without them, quality suffers. Designs must be reviewed against requirements. Code must be reviewed against designs. Code must also be reviewed for other quality standards. Design and code reviews can have varying levels of formality. The review process should achieve the following goals:

- Schedule time for developers to look at specific design or code modules.
- Record findings formally so that there is a permanent record of what needs to be changed.
- Schedule time for developers to make the changes to the design and code artifacts.

Reviews are held regularly as per the iterative approach. This keeps developers focused and corrects problems early. Regular reviews also keep reviews fresh by reviewing smaller chunks of code. Given the amount of code, reviewing all of it at the end of implementation can be monotonous, and the quality of the inspection reduced.

End-to-end life cycle

The WebSphere Application Server V7 environment and its tie-in to other Rational tools offers the developer support at every stage of the application development life cycle.

Key stages in this life cycle are as follows:

- Requirements gathering and analysis
- Prototyping
- High-level design
- Low-level design
- Implementation/coding/debugging
- Unit testing
- Integration testing
- Functional verification testing
- Independent testing
- Acceptance testing
- Performance testing
- Deployment
- Maintenance (including fixes, modifications, extensions)

Software development life cycle key concepts

This section looks at the fundamental concepts in the software development life cycle. Trends in the industry have moved from a static waterfall model to a more dynamic iterative model. Within these models, quality has to be assured by a process of verification and validation.

- Waterfall model

  The life cycle can be looked at as a waterfall model because the output of each stage spills into the subsequent stage. For example, when the high-level design has been created, the artifacts created during that stage (for example the design of a user interface) feed into the low-level design of the modules needed to implement that user interface. This way of looking at the application development life cycle is largely non-iterative. This means that each stage is started when another stage finishes and there is little or no overlap.

- Iterative model

  An iterative life cycle model addresses the same stages as the waterfall model but there is a degree of overlap. *Iterative* means that the cycle can feed back into itself and that software grows as the life cycle is repeated.

  This iterative behavior occurs at a macro and micro level. At a macro level, the entire life cycle repeats itself. The maintenance stage often leads back to the requirements gathering and analysis stage. At a micro level, the review of one

stage might lead back to the start of the stage again or indeed back to the start of another stage.

For example, if during low-level design, a flaw or inconsistency is found in the high-level design, a micro HLD stage is followed again to ensure that the HLD is updated correctly. Indeed, this might even feed back to a micro cycle of the requirements gathering and analysis stage if the flaw in the HLD was found to be due to vagueness in the requirements analysis.

- Verification and validation

  The micro iteration approach involves the concepts of verification and validation (V&V). Verification means to prove something is correct. Validation means to prove something is sound and logical. In software and systems development the two words map respectively to two key questions:

  – Are we building the right system?

    This is the verification question. Is the system that we are building correct in respect to its requirements? This question can be answered by referring to the requirements analysis and design documentation. If anything is not clear or is inconsistent from these stages, then these stages require a further iteration.

  – Are we building the system right?

    This is the validation question. Is the system soundly and logically constructed? Is it robust? Is it maintainable? Does it conform to standards? This question can be answered with varying degrees of formality. For example, the logic of an algorithm might be reviewed informally by a development team. At the other extreme, the logic of an algorithm might be formally proved using mathematical methods (such as the mathematical formal method language Z).

  Quality assurance in software development is achieved by a process of V&V in the development life cycle. This involves a cyclical review process where something is produced and is either validated or verified. This process is iterated upon until a satisfactory level of quality is achieved.

The Rational Unified Process (RUP)

Rational software development tools are all built around the Rational Unified Process (RUP) to a degree. RUP is an iterative software development process. It is iterative at a macro and micro level At the macro level, phases of Inception, Elaboration, Construction, and Transition can be identified in the process. These phases are basically periods of initial planning, more detailed planning, implementation, and finalizing and moving on to the next project cycle.

The next cycle repeats these phases. At the micro level, each phase might go through several iterations of itself. For example, during a construction phase, coding, testing, and re-coding might take place a number of times. Figure A-2 gives an overview of RUP.

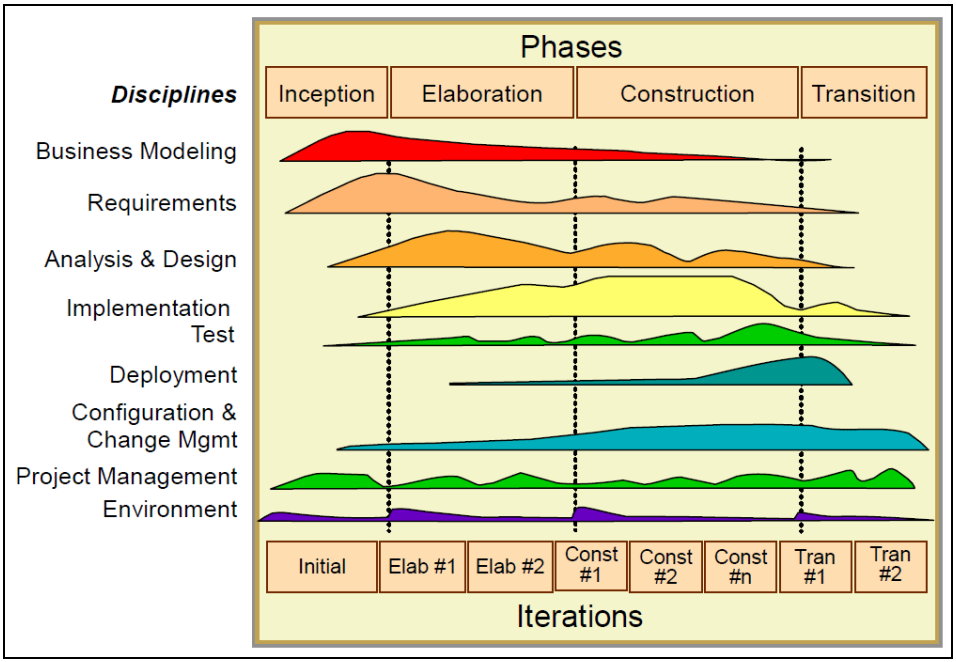


Figure A-2 Rational Unified Process overview

As also shown in Figure A-2, RUP identifies a number of disciplines that are practiced during the various phases. These disciplines are practiced during all phases but the amount of activity in each phase varies. Clearly, the requirements discipline is more active during the earlier inception and elaboration phases, for example.

RUP maps disciplines to roles. There are many roles but the roles break down into four basic sets of roles:

- Analysts
- Developers
- Testers
- Managers

Members of the team might take on more than one role. More than one team member might have the same role. Each role might require the practice of more than one discipline. RUP can be followed without using Rational Software. It is just a process specification. However, RUP provides specific guidance (called

Tool Mentors) on how to use Rational Software when following the process. The disciplines identified in RUP (such as requirements analysis, design or testing) map to specific pieces of Rational software and artifacts that this software generates. RUP is a process that can be employed as much as required.

Note: For more information about RUP, visit the following web page:

http://www.ibm.com/software/awdtools/rup

Agile development

Agile development is a collaborative, incremental, and iterative approach to software development that can produce high-quality software in a cost effective and timely manner. Unlike traditional software development, agile development emphasizes flexibility, continuous testing and integration, and rapid delivery of functionality. By focusing on collaboration and iteration, agile development teams demonstrate the following skill sets:

- Increased team efficiency
- Lower development costs
- Faster time to market
- Better quality software
- Systems better suited to customer needs
- Repeatable results

Teams can adopt and improve agile development strategies for their unique needs and challenges. Agile solutions includes agile practices, process improvement services, training and mentoring services, and products to help companies adopt and tailor agile strategies to meet the unique challenges faced by software delivery teams. For more information about the Agile development, visit the following web page:

http://www.ibm.com/software/awdtools/rup

Java EE best practices: For more information about Java EE best practices refer to IBM WebSphere Developer Technical Journal article: *The Top Java EE best practices*, available from the following web page:

http://www.ibm.com/developerworks/websphere/techjournal/0701_botzum/0701_botzum.html

Also, see the IBM WebSphere Developer Technical Journal article: *Migrating to IBM WebSphere Application Server*, available from the following web page:

http://www.ibm.com/developerworks/websphere/techjournal/0107_beaton/beaton.html

B

Migration questionnaires

This chapter contains sets of questions that you can use to build your own questionnaire based on the unique challenges you foresee. It is not intended to be a final questionnaire, but a base from which you can learn how migration questionnaires are built. Other specific questionnaires might be required. (such as SOA and web 2.0 questionnaires). You can also see 4.3.2, “Getting help” on page 77 to get help from IBM.

© Copyright IBM Corp. 2010. All rights reserved.

Business requirements

Business requirements questions are as follows:

1. What is motivating the business to make this change?
2. Is the application business critical?
3. What is the target version of WebSphere Application Server?
4. What other WebSphere products are involved in this migration?

   WebSphere Portal Server and WebSphere Edge Server possible are examples.
5. By what date must this application migration be complete?
6. What is the risk to the business if the change cannot be implemented by that date?
7. What is the risk to the business when the application is not running?
8. Is this migration part of a new application release/version?
9. Please describe any important milestones for the migration. Include dates for development, QA, and so forth.
10. How do you describe a successful migration?

General information

General information questions are as follows:

1. What is the name of the application?
2. What URL is used to access the application?
3. What is the current version of the application?
4. What is the audience for the application: internal, external, or both?
5. What is the primary function of the application?
6. What services does this application provide?
7. Who are the users of the application?
8. Describe any upgrades that are considered part of this migration?

   This might include such things as platform infrastructure (OS or related stack components) or exploitation of features provided by Java EE.
9. Can the migration be partitioned?

10. Can the application be divided into smaller pieces that can individually be migrated?

    Staged migration of the database, JSP, EJB, and so forth, enables incremental migration, providing greater opportunity for quality control and isolation of unexpected problems.

11. How frequently do you schedule an independent review of your application code?

Application architecture

General application architecture questions are as follows:

1. Does the application architecture conform to the Model 2 architecture prescribed by Java EE?
2. Describe the application's use of established design patterns.
3. Does the application, for example, employ well-known patterns such as Facade, Command, Strategy or Proxy?

Dependencies

Dependency questions are as follows:

1. Are there dependency cycles between packages?

   A dependency cycle results when two or more types are involved in a cycle of references that binds them tightly to one-another. Dependency cycles are not always bad. Cycles are often desired between two or more types. Dependency cycles become a problem, however, when they exist between packages or modules. The danger is that they introduce a tight coupling between those packages or modules, which makes the code generally more brittle and difficult to change. The point of this question is to understand if any such cycles exist between packages.

2. Are there dependency cycles between Java EE modules?

   A common dependency cycle between Java EE modules involves an EJB module that makes reference to helper classes in a Java utility module that, in turn, makes references back to the EJB module. Such helper classes are typically used as convenience methods for finding EJB homes, for example.

3. Describe any classloading dependencies in the application.

   Class loading issues and dependency to libraries are common in older Java EE-style applications. The earlier versions of the specifications did not state clearly how the class loading has to happen and how the library classes need to accessed.

Persistence

There is one question related to persistence:

1. How is persistence of application data implemented?

   This question is concerned with understanding how application data is made persistent. Typically, application data is stored in a database and a layer in the application is responsible for moving that data from the database into Java objects and back again. Persistence might be implemented using EJB CMP entity beans, a third-party product such as TopLink for Java, or a custom implementation of a Data Access Object (DAO) pattern. It might also be implemented in an ad hoc fashion using JDBC statements throughout the application code.

National language

National language questions are as follows:

1. What national languages are supported by the application?
2. How is language determined by the application?

   The application might use the language attribute from the HTTP Request Header, or perhaps profile information provided by a user using a combination of cookies and HTTP session data.

3. How is language data stored by the application?

Code

This section has questions related with the code. Approximate answers will suffice for these questions. However, accurate answers yield more accurate results

Java

Java questions are as follows:

1. How many custom classes are defined in the application?

   A custom class is one that was created by your development team. Do not include classes that are part of the standard Java libraries, middleware, or third-party supplied libraries. Also, do not include classes generated by the container to support your application in run time.

2. How many custom classes in the application are declared as abstract?
3. How many custom interfaces are in the application?

   A custom interface is one that was created by your development team. Do not include interfaces that are part of the standard Java libraries, middleware, or third-party supplied libraries.

4. How many lines of code are in the application?

   Provide an indication of useful lines of code (exclude comment lines from this value).

EJB usage

EJB usage questions are as follows:

1. What version of the EJB specification is used?
2. Do all the enterprise beans in the application conform strictly to this version of the specification?
3. How many Stateless Session beans are in the application?
4. How many Stateful Session beans are in the application?
5. How many CMP Entity beans are in the application?
6. How many BMP Entity beans are in the application?
7. How many Message Driven beans are in the application?

Servlets and JSPs

Servlets and JSP questions are as follows:

1. What version of the JSP specification is used?
2. Do all the JSP files in the application conform strictly to this version of the specification?
3. How many JSPs files are in the application?
4. Do your JSP files contain Java code?

   List the third-party tag libraries (taglibs) used by the application. Include version numbers if possible.
5. Describe any custom tag libraries (taglibs) you have constructed for the application
6. What version of the servlet specification is used?
7. Do all the servlets in the application conform strictly to this version of the specification?
8. How many servlets are in the application?
9. Approximately how much data is maintained in the HttpSession?

   This can be difficult to estimate, especially if the size the session data varies widely. One possible way to get an accurate measure of a session at a particular moment in time is to build a servlet that uses an ObjectWriteStream to dump the contents of the HttpSession to a file. The size of the file is a good approximation (though not absolutely correct measure of the session size.
10. Are all objects that are stored in the HttpSession serializable?

    The application server uses serialization to pass data between servers (JVMs). An object is serializable if it implements the java.io.Serializable interface.

    Answer Yes to this question only if all objects are serializable. This includes objects that are indirectly included in the HttpSession through references from other objects.
11. Describe any issues that might prevent sharing data your application stores in the HttpSession across servers in a cluster.

Web services

Web services questions are as follows:

1. Enumerate the external web services your application accesses.

   By *external*, we mean external to the application. This includes any web services that are hosted by other applications regardless of the language or platform they are running on. If possible, provide a description of each web service used and note any potential issues that you are aware of that might affect the migration.

2. Enumerate the internal web services your application accesses.

   An *internal* web service is one that is part of your application. Your application might, for example, use a web service to implement the Facade pattern between your servlet and EJB layers.

3. Enumerate the web services your application implements.

   For this answer, list the web services that your application provides for internal or external clients. If possible, provide a description of the function that each web service provides.

4. Are the web services implemented by your application WS-I compliant? If so, indicate the WS-I version.

Database access

Database access questions are as follows:

1. What version of the JDBC specification is used?
2. Does the use of JDBC conform strictly to this version of the specification?
3. How many distinct JDBC queries does the application make?
4. What database managers are used by the application?
5. Do the JDBC queries make use of vendor-specific SQL extensions?
6. Does the application make direct use of DriverManager?
7. Does the application make Use of J2EE connection pooling through data sources?
8. Describe any third-party connection pooling mechanisms in use.
9. Does the application make use, for example, of Oracle Connection Pooling?
10. Are all JDBC connections explicitly closed?
11. Are all JDBC statements explicitly closed?
12. Are all JDBC resources closed inside `finally` clauses?

    Code contained in a `finally` block of an exception handler is guaranteed to run (whether an exception occurs or not). By closing these shared resources in a `finally` clause, you ensure that these resources can be kept available.

JMS

JMS questions are as follows:

1. What version of the JMS specification is used?
2. Does the use of JMS conform strictly to this version of the specification?
3. How many queues does the application make use of?
4. How many topics does the application make use of?
5. What kind of data is passed through JMS?

   For example: "plain text", "XML", or "serialized objects"
6. Does the application write to JMS queues?
7. Does the application read from JMS?
8. Does the application do blocking reads from JMS?
9. Does the application do blocking reads from JMS in Session or Entity beans?

JNDI naming

JNDI naming questions are as follows:

1. Does the application use JNDI to access resources other than EJB Homes?
2. Do the application's deployment descriptors contain local JNDI resource references?

   JNDI resource references sections are within web.xml and ejb-jar.xml.
3. Describe a typical JNDI access from the application. Provide a code snippet if possible.
4. Are all JNDI resources (InitialContexts) closed inside `finally` clauses?

   Code contained in a `finally` block of an exception handler is guaranteed to run (whether an exception occurs or not). By closing these shared resources in a `finally` clause, you ensure that these resources can be kept available.

Application trace and logging

Application trace and logging questions are as follows:

1. What technology is used by the application to provide logging?

   The application might use, for example, a third-party product such as Log4J. Alternatively, the application might write messages to standard out (System.out).

2. Is logging used consistently throughout the application?

Struts

If the application does not make use of Struts, skip this section. Struts questions are as follows:

1. What version of Struts is used?
2. How many custom ActionForms are in the application?
3. How many custom ActionHandlers are in the application?
4. Describe any modifications you have made to Struts.
5. Struts is an open source framework and is customized by organizations to provide extended functionality. If you have extended Struts, describe that extension.

Transactions

Transactions questions are as follows:

1. Are transactions managed implicitly or explicitly?
2. Does the application require two-phase commits (2PC)?
3. What transactional systems does the application interact with?

Threads

This section is concerned with the application's direct manipulation of threads. This section is not concerned with threads managed by the application server.

Thread questions are as follows:

1. Does the application create threads?
2. Does the application create daemon threads?
3. Does the application explicitly stop threads using the stop() method?
4. List any third-party thread pool libraries used by the application. Include version numbers if possible.
5. Describe any custom thread pool libraries used by the application.
6. Describe how the application makes use of threads.

   This question is concerned with how your application makes explicit use of threads. For example, does your application spawn threads to perform work in parallel, or to provide timeouts for various services. If it is the latter, indicate what services the threads are providing timeouts for.

Sockets

Skip this section if your application does not make direct use of sockets. This section is concerned with sockets created and used directly by your application; it is not concerned with sockets managed by the application server.

Socket questions are as follows:

1. Describe how the application makes direct use of sockets.
2. Does your application create any daemon sockets?
3. Does your application use secure sockets (SSL)?

XML

XML questions are as follows:

1. What XML libraries does the application use? Include version information.
2. What XSLT libraries does the application use? Include version information.

Development migration questionnaire

The following questions are related to development environment for the migration.

Workstation configuration

Workstation configuration questions are as follows:

1. What hardware is used?
2. What operating system and version is used?
3. How much RAM is installed?

Integrated Development Environment

Integrated Development Environment questions are as follows:

1. What Java Integrated Development Environments (IDE) are used?
2. What source code management (SCM) tool is in use?
3. How are source directories organized?
4. What tools are used for unit testing?
5. What other tools do you use as part of your development process?
6. Will WebSphere development tools be adopted as part of this migration effort?
7. Which development tools are adopted as part of this migration?
   - WebSphere Studio Application Developer
   - WebSphere Studio Application Developer Integration Edition
   - Rational Application Developer
   - Other

Development test configuration

Development test configuration questions are as follows:

1. Is dedicated development test hardware available for the development teams?
2. What hardware is used?
3. What operating system and version is used?
4. How much RAM is installed?
5. What application server and version is used for development testing?
6. What other software is used for development testing?

Software development skills

Software development skills questions are as follows:

1. How many software developers form the team that built the application?
2. How many of your developers have architect-level skills?
3. How many developers are available to do the migration?
4. How familiar are your developers with Java EE?
   - They are experts with comprehensive knowledge of Java EE
   - They are familiar with the various technologies that make up Java EE (like servlets and EJBs)
   - They are not at all familiar with Java EE
5. Do your developers have any experience using Eclipse for software development?
6. Do developers have experience using WebSphere Studio for software development?
7. Are there specific education requirements for the development team?

Development methodology

Development methodology questions are as follows:

1. What development methodology is used?
2. How long is a typical development cycle?
3. How frequently are internal releases of the application delivered?
4. This includes releases for internal testing, or user acceptance testing.
5. How frequently are external releases of the application delivered?
6. How often are new versions of application delivered into the production environment?

Build and packaging

Build and packaging questions are as follows:

1. What tools are used to build the application?

   Examples: Ant, make, or none. Provide version information if possible.
2. In what form are packaged applications delivered to the runtime environment? Are applications packaged as EAR, WAR or JAR files? Or are they delivered in another form?
3. How are packaged applications delivered to the runtime environment?
4. What mechanisms are in place to deliver the application to the runtime environment? Is the packaged application delivered through FTP, or by another transport mechanism?
5. What process is followed to deliver the application?
6. How is static content packaged and delivered to the runtime environment?
7. Are installation and configuration scripts included with the packaged application?

Ant

Skip this section if you do not use Ant for builds.

1. What version of Ant is used?
2. Does your build process use one or many Ant scripts?
   - One
   - Many
3. What optional Ant tasks do you use as part of your build?

Runtime migration questionnaire

The following questions are runtime migration-related:

General

General questions are as follows:

1. Will the existing runtime infrastructure be migrated?
2. Will you have to support the existing runtime infrastructure after the migration?

   Answer yes if you expect to migrate only your current applications and will leave them running on the current infrastructure.
3. Does your runtime infrastructure include a dedicated pre-production staging environment?

   Your pre-production environment might be called the Quality Assurance environment. Regardless of the given name, this environment is used to test the overall quality of your environment and applications prior to moving those applications into the production environment.
4. Does your runtime infrastructure include a dedicated performance testing environment?

   Describe the separate runtime environments (production, pre-production, system test, development test, performance test, etc.) that are included in your runtime infrastructure.

Current hardware

Questions about current hardware are as follows:

1. Describe the hardware used to run your current application server.

   If your environment is heterogeneous, describe each hardware configuration. Indicate the quantity of each configuration. Also indicate the operating system and version currently in use.
2. How many production machines currently host the production application?

   Describe any additional hardware used in the production environment. Include all the details from firewalls, IP sprayers, web servers, clustering manager, database servers, firewall, intrusion detection systems, and so forth. Indicate software in use (if any) on these systems, including operating system and version.

3. Describe how the configuration of the pre-production environment differs from the configuration of the production environment?
4. Is the pre-production environment configured with a similar complexity, but at a smaller scale? Are there key elements missing from the pre-production environment?
5. Describe the hardware you plan to use to run the migrated applications.

   If your environment is heterogeneous, describe each hardware configuration. Indicate the quantity of each configuration. Also indicate the operating system and version currently in use.
6. How many production machines are used to host the production application?

   Describe any additional hardware used in the production environment. Include all the details from firewalls, IP sprayers, web servers, clustering manager, databases, firewall, intrusion detection systems, and so forth. Indicate software in use (if any) on these systems, including operating system and version.

Software

Questions about software are as follows:

1. Which version of Java Runtime Environment (JRE) do your current production hosts use?
2. Provide the name of the provider of the JRE.

   This might be Sun, IBM or another JRE vendor.
3. Describe other software that is installed and used on the same production hardware as the application servers.
4. Describe the infrastructure of the host bridge network if the application employs the connection services to host systems (for example, CICS).

HTTP Server

HTTP Server questions are as follows:

1. What HTTP Server is used?
2. Is the HTTP Server configured to provide affinity to a particular application server instance?
3. How many servers will host the web servers?

   IBM WebSphere is capable of working with almost all web servers in the market today. It is important to understand how the current web server is being used and what is the impact if WebSphere Application Server continues to use the same web server during the production switch
4. Describe any other services provided by the HTTP server. This includes services beyond serving static content and fronting for the application server. Does your HTTP server, for example, provide additional functionality using CGI/Perl?
5. Describe the infrastructure for secure network propagation.
6. Enabling HTTPS on the web server is one way of doing it from the browser to the web server end. Enterprises also use hardware accelerators to improve the speed of the SSL encryption and the decryption process. If so how does it integrate with the web server?
7. Is the communication from the HTTP server plug-in to the application server secure in the current environment?

Network edge

Network edge questions are as follows:

1. What load balancing technologies are used?
2. Is the load balancer configured to provide affinity to a particular HTTP server instance?
3. Describe how failover is provided for your load balancer.

Availability

Availability questions are as follows:

1. Describe any existing service level agreements concerned with the availability of the applications.

   This determines the availability and the amount of flexibility to switch to was run time.

2. Describe how clustering is provided in the current environment. Explain about your current clustering environment.

3. Describe the process for the production release rollout of an application?

   If the process differs for various applications, describe each of the processes.

4. What is the current backup strategy when a node or server fails or breaks down?

   Explain in detail your fail over strategy.

Rollout issues

Questions about rollout issues are as follows:

1. Will old applications need to coexist with newly migrated applications?

2. Does the migration include all the applications on the production server? If not, is there a plan to migrate those applications as well?

   Co-existence is important especially when migrating common frameworks and application development models into a new platform. There can be application dependencies that needs to be maintained when moved into the new platform.

3. Are there any other software updates scheduled to be done to the production machines during the migration process?

   Migration of the production environment is a important task and mitigation of risk is the key during the migration. Hardware and software updates are only recommended when they are required as part of the migration. It is advised that they be decoupled to keep the number of variables to minimum.

Administration

Administration questions are as follows:

1. How many administrators do you have to support the production environment?
2. Do you have dedicated administrators for your production infrastructure?
3. Describe the skill level and experience of the administrators of the production environment.
4. How do you currently deploy the applications?
5. Do the production administrators use deploy scripts to accomplish this task? Does it involve custom scripting? Explain the process.
6. Describe any custom scripts that are used for administering the application servers.
7. Do you use any custom scripts for administering the current set up? If so what they used for?
8. Is the production environment set up to take care of log rotation and file system maintenance?
9. Log files can become really large and can fill up the available space within the production host. Is there a procedure or a process in place to handle file system maintenance?

Security

Security questions are as follows:

1. How important is security in your production runtime infrastructure?
   - Critical
   - Very important
   - Important
   - Not important
2. Describe how internal security of the application server is managed.

   There are several roles applicable within the production environment, including root administrator, machine maintenance operator, application deployer, and developer. Explain how access to the application server hardware and software is managed with respect to these roles.
3. How are internal security policies enforced?
4. How are users authenticated?
5. How are user authorizations managed?

6. Describe your use of application-specific services for user authentication and authorization.
7. Describe your use of custom services for user authentication and authorization.
8. Describe any third-party security mechanisms in use.

   This might include products such as Netegrity Siteminder.
9. Describe how your production runtime provides (or participates) in a single sign-on solution. Describe any network security measures in place.
10. Are firewalls positioned between tiers in your configuration? If so, how and where are they configured?

Testing migration questionnaire

The following questions are for the testing environment:

Hardware

Questions about the hardware used are as follows:

1. Is dedicated hardware available specifically for user acceptance testing?
2. Is dedicated hardware available specifically for quality assurance testing?
3. Is dedicated hardware available specifically for performance testing?
4. Describe how the testing hardware is different in configuration from the production hardware.

Practices and tools

Questions about the practices and tools used are as follows:

1. What kinds of testing are done and when?
2. What is the typical length of the regression test cycle?
3. How frequently do applications undergo regression testing?
4. What is the typical length of the user acceptance test cycle?
5. How frequently do applications undergo user acceptance testing?
6. Is User Acceptance Testing required as part of this migration effort?
7. What is the typical length of the quality assurance test cycle?

8. How frequently are do applications undergo quality assurance testing?
9. Describe the testing stages that an application is taken through when moving from the development environment into production.
10. How do you test applications for compliance with the J2EE specification?
11. Do you have an established set of application development best practices? This includes coding standards and code maintenance and performance best practices.
12. How are application development best practices enforced?
13. What tools do you use to unit test your application code?
14. What tools do you use to function test your application code?
15. What tools do you use to regression test your application code?
16. What tools do you use to performance test your application code?
17. Does the testing procedure contain any harness whose code is dependent on the source platform?

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247870

Alternatively, you can go to the IBM Redbooks web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247870.

© Copyright IBM Corp. 2010. All rights reserved.

Using the web material

The additional web material that accompanies this book includes the following files:

File name	*Description*
trade3-WebLogic.zip.zip	Compressed Trade 3.1 sample source code and binaries.
xpetstore-cmp-wls.data.zip	Compressed SQL with DB2 formatted data for loading the xPetstore EJB database when deploying and migrating from Oracle WebLogic Server 9.2.
CalculatorWS - NetBeans Projects.zip	Compressed CalculatorWS application that runs on GlassFish. This is a sample application written by the residency team to show how to migrate web services applications to WebSphere Application Server,
CalculatorWS - RAD Projects.zip	Compressed CalculatorWS application. This is the version that was migrated to the WebSphere Application Server. This is a sample application written by the residency team to show how to migrate web services applications to WebSphere Application Server.

System requirements for downloading the web material

The following system configuration is recommended:

Hard disk space:	40 GB minimum
Operating System:	Windows XP Professional
Processor:	Pentium IV 3.0 GHz or higher
Memory:	2 GB minimum

How to use the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material compressed file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 380. Note that the documents referenced here may be available in softcopy only.

- *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690
- *Migrating WebLogic Applications to WebSphere Advanced Edition*, SG24-5956
- *Migrating WebLogic Applications to WebSphere Advanced Edition V4*, SG24-6179
- *WebSphere Application Server V7: Concepts, Planning and Design*, SG24-7708
- *Migrating Applications from WebLogic, JBoss and Tomcat to WebSphere V6*, SG24-6690
- *Experience JEE! Using Rational Application Developer V7.5*, SG24-7827
- *MySQL to DB2 Conversion Guide*, SG24-7093
- *WebSphere Application Server V7 Messaging Administration Guide*, SG24-7770
- *Getting Started with WebSphere Application Server Feature Pack for Service Component Architecture*, REDP-4633
- *Migrating WebLogic Applications to WebSphere V5*, REDP-0448

© Copyright IBM Corp. 2010. All rights reserved.

Online resources

These web sites are also relevant as further information sources:

- General WebSphere information:

  http://www-01.ibm.com/software/websphere/

- WebSphere Application Server V7.0 information:

  http://www.ibm.com/software/webservers/appserv/was/

- WebSphere Application Server for Developers V7.0 information:

  http://www.ibm.com/developerworks/downloads/ws/wasdevelopers/

- WebSphere Application Server - Express V7.0 information:

  http://www.ibm.com/software/webservers/appserv/express/

- WebSphere Application Server Network Deployment V7.0 information:

  http://www.ibm.com/software/webservers/appserv/was/network/

- WebSphere Application Server for z/OS V7.0 information:

  http://www.ibm.com/software/webservers/appserv/zos_os390/

- WebSphere Application Server for z/OS V7.0 information:

  http://www.ibm.com/software/webservers/appserv/zos_os390/

- Application Migration Tool download location:

  http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html

- Eclipse IDE download location:

  http://www.eclipse.org

- “What’s new in WebSphere Application Server V7?” white paper:

  http://www.ibm.com/developerworks/websphere/library/techarticles/0809_alcott/0809_alcott.html

- WebSphere Application Server Feature Packs information:

  http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/

- IBM Software Services for WebSphere (ISSW) information:

  http://www3.software.ibm.com/ibmdl/pub/software/dw/wes/pdf/services/DevelopDeployFinal.pdf

- WebSphere education information:

  http://www-01.ibm.com/software/websphere/education/

- IBM WebSphere Application Server V7 Information center:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/welc_howdoi_tprf.html

- IBM DB2 Information center:

  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp

- IBM DB2 Universal Database Express Edition download location:

  http://www.ibm.com/developerworks/downloads/im/udbexp

- Java Platform, Enterprise Edition 5 (Java EE 5) Specification:

  http://jcp.org/en/jsr/detail?id=244

- Apache Tomcat 6.0.26:

  http://tomcat.apache.org/tomcat-6.0-doc/index.html

- Migrating a Microsoft Access 2000 Database to IBM DB2 Universal Database 7.2:

  http://www.ibm.com/developerworks/data/library/techarticle/alazzawe/0112alazzawe2.html

- GlassFish licensing information:

  https://glassfish.dev.java.net/public/CDDL+GPL.html

- Java SDK 6 download site:

  http://java.sun.com/javase/downloads/index.jsp

- GlassFish Application Server download site:

  https://glassfish.dev.java.net/public/downloadsindex.html

- Apache Maven website:

  http://maven.apache.org/

- Apache Maven installation instructions:

  http://maven.apache.org/download.html

- Seam website:

  http://seamframework.org/

- PMD Source Code Analysis tool

  http://pmd.sourceforge.net

- WebSphere migrations: Migrating WebLogic server and application configurations to WebSphere Application Server

  http://www.ibm.com/developerworks/websphere/library/techarticles/0706_vines/0706_vines.html

- Application Verification KIT (AVK)

  http//java.sun.com/j2ee/verified/avk_enterprise.html

- Set up WebLogic test server in RAD

  http://www.ibm.com/developerworks/rational/library/06/1121_tarte/index.html?ca=drs

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, and order hardcopy Redbooks publications, at this web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols
39

Numerics
2-phase commit transactions 104

A
abstraction 327
administrative agent 47
administrative scripting 4
Ajax Development Toolkit 63
Ajax implementations 63
Ajax(Asynchronous JavaScript and XML) 61
analyzing migration problems 211, 240, 261
Ant 101, 200
Ant build script 166
Apache Log4j 17
Apache Maven 201
Apache Software Foundation 276
Apache Tomcat 278
Apache Tomcat 6.0.26 2
Apache Wink 62
application class loader 10
application infrastructure 28
application integration 29
application migration 82
Application Migration Tool
  analyze 120
  analyze entire workspace 118
  analyze last launched 129
  Class-Path Review tab 120
  configuring and running 116
  downloading 95
  fixing errors reported by 122
  installing 95
  Java code review 116
  Java Code Review tab 120
  JBoss migration 220
  JSP Code Review tab 120
  overview 103
  plug-in 95
  Quickfix All 132
  rerunning 129
  rule settings 119
  rules sets 119
  running 182
  scope tab 118
  Software Analyzer 118
  XML File Review tab 120
application packaging 15
application portability 15
applications covered in this book 3
ASF 276
AUTO_INCREMENT columns 19
automated tests 328
Available WebSphere Application Server Feature Packs 55

B
Bean-Managed Persistence (BMP) EJB 4
BM ILOG Core Products & Technologies 30
bootstrap 9
business process management 29
business rule management systems 29

C
CalculatorWS 318
classloader modes 11
classloaders 9
ClassNotFound 9
CLASSPATH 9
cleanup phase 81
Cloudscape 42
CMP EJB 2.0 104, 166
compile time dependencies. 240
configuring
  JMS 146, 188
  Session Beans and Container Managed Beans 194
  the datasource in WebSphere 294
container-managed persistence (CMP) 13
container-managed transactions 337
Customer Information Control System (CICS) Transaction Server 28

© Copyright IBM Corp. 2010. All rights reserved.

D

Data Access Objects (DAO) 342
database schema 13
database servers 42
database-related issues 19
DB2 JDBC driver 168
deploying Java Pet Store on GlassFish 305
deployment descriptors 12, 303
deployment manager 51
design patterns 341
development and deployment tools 63
development environment issues 26
Differences in Java EE implementations 8
Directory servers 43
Dojo Foundation 63
Dojo project 63

E

Easy JSP Forum application 290
  downloading 290
Easy JSP Forum migration 289
Eclipse IDE 95
  downloading 95
EJB module 16
EJB to RDB mapping tool 19
EJBHomeFactory 342
enable session persistence 338
enterprise java beans (EJB) 4
enterprise java beans (EJBs) 31
enterprise resource planning (ERP) 36
Entity and Stateless Session EJB 2.0 104
Evolving Java application development standards 38

F

factory design pattern 343
Feature Pack for XML 56
feature packs 55
Federated Repository Configuration 43
FileNet Business Process Manager 29
fixing
  Hibernate configuration 263
  JNDI lookup 212, 243
  persistence configuration 245
  persistence settings 216
  runtime dependencies 212
  the web module dependencies 211

G

generating Seam application using seam-gen 230
getting help 77
GlassFish 2.0 2
GlassFish migration 301
  CalculatorWS 318
  dependency injection 303
  deploying Java Pet Store on GlassFish 305
  downloading 304
  installation 304
  Installation and configuration 304
  Java Pet Store 2.0 Reference application 303
  licensing information 302
  major issues 322
  prerequisites and assumptions 302
  Web Services application 318
GNU General Public License (GPL) 279, 302
graphical user interface (GUI) 36
GUI testing 329

H

HSQL 19

I

IBM Business Partner 77
IBM Business Partner Technical Strategy and Enablement (BPTSE) 77
IBM DB2 Server Express Edition 97
IBM DB2 UDB 8.2 101
IBM ILOG ODM Enterprise 30
IBM ILOG OPL-CPLEX Development Bundles 30
IBM POWER® family of processors 39
IBM Rational Application Developer for WebSphere Software 88
IBM Rational Application Developer V6 102
IBM runtime environment 30
IBM Software Services for WebSphere (ISSW) 77
IBM System i (iSeries®) 39
IBM System z processors 39
IBM Tivoli Directory Server 43
IBM WebSphere 27
IBM WebSphere Application Server Feature Packs
  development and deployment tools 63
  IBM WebSphere Application Server Feature Pack for Web 2.0 61
IBM WebSphere Application Server V7 101
IBM WebSphere Developer Services 78
IBM WebSphere Platform 28

IBM z/OS Integrated Security Services 43
IBM z/OS.e Security Server 43
implementation considerations 79
implementation phases 80
importing HAR contents in source form 270
importing the application EAR file 175
importing the application source code 177
importing the EAR file and source code to RAD 205, 235, 258
importing the Trade EAR file 114
installing
  Apache Tomcat 278
  Application Migration Tool V1.1 113
  DB2 Universal Database Express Edition 98
  IBM Installation Manager 89
  IBM Rational Application Developer for WebSphere Software 88
    installing the latest fixpack 90
  JBoss Application Server 202
  JBoss Application Server Version 4.2.3 203
  JBoss Application Server Version 5.1.0 202
integrated test environment 253, 269
Intel EM64T or AMD™ Opteron™ 39
Intel Pentium® processor 39
Intel® Itanium® 2 processor 39
intelligent provisioning 46
interoperability and integrations 74
iterative model 350

J

J2EE 2
J2EE 1.4 to Java EE 5 migration considerations 21
J2SE 1.4 101
Jakarta Commons Logging V1.2 12
JAR (Java ARchive) 15
Java BluePrints web site 303
Java Database Connectivity (JDBC) 4
Java DataBase Connectivity (JDBC™) API 38
Java Development Kit (JDK™) 6.0 37
Java EE application clients 20
Java EE application server compatibility 8
Java EE Connector Architecture (JCA) 4
Java Mail 4
Java Management Extensions (JMX) 4
Java Naming and Directory Interface (JNDI) 120
Java Native Interface (JNI) migrating 18
Java Persistence API (JPA) 4, 38, 45
Java Pet Store Reference application 303
Java Platform Enterprise Edition (Java EE) 36
Java Platform Enterprise Edition (Java EE) 36
Java Platform name changes 36
Java SE and EE technologies covered 4
Java Server Faces (JSF) 22, 333
Java Server Page (JSP) 22, 104, 166
  taglibs 4
Java Servlet 2.5 (JSR 154) 38
Java source code 18
Java-based applications 30
JavaDoc 329
JavaScript Object Notation (JSON) 62
JavaServer™ Pages (JSR 245) 38
JBoss 4.2 and JBoss 5.1 2
JBoss Application Server 202
JBoss migration
  Configuring precompiled Java archive (JAR) libraries 207
  creating and populating the database 256
  creating the datasource 247
  fixing the problems 211
  JBoss Application Server installation 202
  JNDI lookup 212
  JSF-EJB3 sample application 204
  managing runtime dependencies 212
  migrating a Seam application 226
  migrating JSF-EJB3 Sample Application from JBoss 203
  migrating Online Brokerage application 254
  migration approach 204, 227
  persistence settings 216
  preparing the environment 200
  running the application 222
  specifications level migration 222
  verifying generated seam-gen application 234
  web module dependencies 211
  WebSphere built-in JPA provider 224
JBoss Seam Framework 203
JCL 12
JDBC 2.1 104
JDBC resources 25
JDom 12
JMS 21
JMS 1.0.2 104
JMS resources 24
Job manager 50
JUnitEE 166

K
Kerberos 55

L
layered application architecture 327
LDAP Server 43
Local Area Networks (LANs) 51
local EJB interface 16
Log4j.jar 17
Lotus Domino Enterprise Server 43
LTPA WSSecurity Default 53

M
mapping tool 19
MDB (message-driven bean) 120
Message Driven Beans 2.0 104, 166
messaging 53
metadata 52
Microsoft Access (*mdb) 290
migrate phase 80
migrating a Seam application generated with seam-gen 226
migrating Web Services applications 322
migration
  considerations 73
  implementation 79
  problems 17
  process 2–3
  process overview 70
  to WebSphere Application Server 306, 320
  Tomcat 289
  WebLogic 169
migration assessment 75
migration assessment report 76
migration issues 3
migration planning 77
migration project plan 78
migration questionnaire review meeting 75
migration questionnaires 75
migration tools 2
MvnForum application 280
  downloading 280
MvnForum migration 279
MySQL 19

N
NetBeans installation file 319
Novell eDirectory 43
Novell® eDirectory 8.7.3 SP9 43

O
Object Oriented (OO) approach 347
offline update 91
Online Brokerage application 255
online update 90
Open JPA 4
OpenJPA 204
OpenLDAP 43
operating systems 40
optimization 30
Oracle WebLogic 9.2 2
OSGi Blueprint service specification 56

P
packaging 33
packaging the application 284
Parallel Sysplex 36
PA-RISC processor 39
partial migration 72
Passport Advantage 78
performance tuning and optimization 84
PetStore 166
PicoContainer 344
plain old Java objects (POJOs) 37
portability 197
portals 30
Portlet 2.0 API 46
post deployment 83
prepare phase 80
preparing database schema 227
primary key generation 104
Processor Value Units (PVUs) 34
progressive disclosure 37
proprietary solutions 326

Q
QoS policies 52, 276
Quick Preview Editor 131

R
Rational Application Developer for Assembly and Deploy V7.5 64
Rational Application Developer for WebSphere Software V7.5 66

Rational Application Developer V7.5 86
Rational Software Analyzer technology 95
Rational Software Delivery Platform (SDP) 66
Rational Unified Process (RUP) 5, 351
Redbooks Web site 380
  Contact us xxiii
Relational Database Synchronization (RDB) 13
Remote EJB interface 16
Resource Adapter archives (RAR) 15
resource definitions 24
RESTful Web services 61
roles and responsibilities 77
Rollout phase 81
rule violation 116
Running CalculatorWS on NetBeans 6.8 and Glass-Fish 3 319
runtime dependencies. 261
runtime migration 83
runtime migration issues 23

S

schema mapping 13
scope of this book 2
Seam application 226
security resources 25
Service Component Architecture (SCA) 4
Service Data Objects (SDO) 38
Service Locator 342
service-oriented architecture (SOA) 27, 61
Servlet 2.3 104, 166
servlets 22
singletons 343
software engineering 341
software installation 277
SPARC workstations 39
Spring Framework 344
SSL WSTransaction 53
Stand Alone LDAP User Registry Configuration 43
standard logging framework 340
standards 44
Stateful Session Beans 2.0 166
Stateless Session Beans 2.0 166
Struts 333
Sun Java Directory Server 43
Sun Microsystems 303
Support from IBM 77
Sybase 42
System z cluster 36
systems management 46

T

test phase 81
thin clients 20
timer and startup services 4
Tomcat migration
  configuring the source environment 280
  creating a new datasource 285
  creating the DB2 database 280, 291
  default connection pool 282
  migrating the Easy JSP Forum application to WebSphere 291
  migrating the MvnForum application 282
  migration approach 279, 289
Trade 3.1 application 100
types of migrations 71

U

undocumented code 329
Unified Modeling Language (UML) 349
use of native code 18
using vendor-specific features 12
utility.jar 17

V

vendor-specific deployment descriptors 14
Verifying JSF-EJB3 sample application 204

W

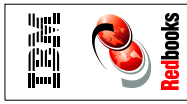
Waterfall model 350
Web 2.0 to SOA Connectivity 62
Web application module 16
Web application module (WAR) 15
Web module class loader 11
Web servers 41
Web Services 22, 45
WebLogic classpath 108
WebLogic migration 178
  configuring the source environment 186
  creating a JMS connection factory 171
  creating a new data source 109
  creating a new database schema 180
  creating a new datasource 156
  creating a new JavaMail session 171
  creating a new JMS server 170
  creating the JMS module 170

creating the JMS queues 170
creating the JMS resources 110
creating the sample application database 107, 168
migrating
Trade Application J2EE Level 143
xpetstore J2EE Specification Level 185
migrating EJBs 158
migrating from older versions 200
migrating the sample application 113
migrating to WebSphere built-in JPA provider (optional) 224
migration approach 105, 166
Oracle WebLogic Server 9.2 99
release date 100
Oracle WebLogic Server 9.2 installation 102
Oracle WebLogic Server 9.2 migration 100
assumptions 101
prerequisites 101
prerequisites and assumptions 101, 276
specifying Web Application References 161
Trade for Oracle WebLogic Server 9.2 migration 104
xPetstore EJB application
downloading 168
xPetstore EJB migration 166
WebLogic migration configuring WebSphere Application Server V7 145
weblogic-cmp-rdbms-jar.xml 141
weblogic-ejb-jar.xml 141
WebSphere 34
WebSphere Adapters 29
WebSphere Application Server
Restarting 162
WebSphere Application Server - Express V7.0 34
WebSphere Application Server - Express V7.0 34
WebSphere Application Server Community Edition 28
WebSphere Application Server Family 28
WebSphere Application Server Feature Pack 55
WebSphere Application Server for Developers V7.0 33–34
WebSphere Application Server for z/OS V7.0 33, 35–36
WebSphere Application Server for z/OS V7.0 35–36
WebSphere Application Server Hypervisor Edition 28
WebSphere Application Server Network Deployment V7.0 35
WebSphere Application Server Network Deployment V7.0 35
WebSphere Application Server V7 30, 34
centralized installation 53
Feature Packs 55
IBM WebSphere Application Server Feature Pack for Communications Enabled Applications 57
IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0 Open Beta 55
more information 30
new opportunities 84
overview 30
packaging 33
platform 28
security 54
supported hardware and software 39
systems management 46
Web services 52
what is new? 43
WebSphere Business Compass 29
WebSphere Business Events 29
WebSphere Business Modeler 29
WebSphere Business Monitor 29
WebSphere Business Services Fabric 29
WebSphere CloudBurst Appliance 28
WebSphere DataPower B2B Appliance XB60 29
WebSphere DataPower Integration Appliance XI50 29
WebSphere DataPower Low Latency Appliance XM70 29
WebSphere DataPower XML Security Gateway XS40 29
WebSphere Dynamic Process Edition 29
WebSphere education 78
WebSphere Enterprise Service Bus 29
WebSphere Extended Deployment Compute Grid 28
WebSphere eXtreme Scale 28
WebSphere ILOG Business Rule Management Systems 29
WebSphere ILOG JRules BRMS 29
WebSphere ILOG Rules for .NET BRMS 29
WebSphere Industry Content Packs 29
WebSphere Integration Developer 29
WebSphere Message Broker 29
WebSphere MQ 29

WebSphere MQ File Transfer Edition 29
WebSphere Partner Gateway 29
WebSphere Portal Server 30
WebSphere Process Server 29
WebSphere Rapid Deployment 67
WebSphere Real Time 28
WebSphere Sensor Events 29
WebSphere Service Registry and Repository 29
WebSphere sMash 28
WebSphere Transformation Extender Industry Packs 29
WebSphere Virtual Enterprise 28
what is not covered in this book 4
which tools to use 68
Wide Area Networks (WANs) 51
Windows Active Directory 43
Windows Active Directory Applications Mode 43
WORA (Write Once, Run Anywhere) 7
WS-Addressing 53
wsadmin 47
WSHTTPS 53
WS-I RSP ND 53
WS-ReliableMessaging persistent 53

X

XDoclet 166
XDoclet tags 26
xPetstore EJB 3.1.3 100



WebSphere Application Server V7: Competitive Migration Guide

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

WebSphere Application Server V7 Competitive Migration Guide

Learn migration strategy and planning

Experiment with the Migration Toolkit

Review migration case studies

This IBM Redbooks publication helps you plan and execute the migration of J2EE applications developed for Oracle WebLogic Server, JBoss, GlassFish, and Apache Tomcat, so that they run on WebSphere Application Server V7.

This book provides detailed information to plan migrations, suggested approaches for developing portable applications, and migration working examples for each of the platforms from which we migrated.

It is not our intention to provide a feature-by-feature comparison of these application servers versus WebSphere Application Server V7, or to argue the relative merits of the products, but to produce practical technical advice for developers who have to migrate applications from these vendors to WebSphere Application Server V7.

The book is intended as a migration guide for IT specialists who are working on migrating applications written for other application servers to WebSphere Application Server V7.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7870-00 ISBN 0738434612